VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Malý grafický LCD displej pro Linux připojitelný přes USB rozhraní

Small Graphics Display for Linux with USB interface

2016 Bc. Miroslav Urbánek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Miroslav Urbánek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Malý grafický LCD displej pro Linux připojitelný přes USB rozhraní
Small Graphics Display for Linux with USB Interface

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit z grafického LCD modulu a mikropočítače samostatnou zobrazovací jednotku připojitelnou přes USB rozhraní k počítači s OS Linux. Modul bude podporovat rozhraní frame-buffer. Využijte stávající projekt FBTFT a v něm nahraďte použité datové rozhraní SPI rozhraním USB.

1. Popište stávající technické řešení používané v projektu FBTFT.
2. Vyberte vhodný modul s grafickým LCD displejem a mikropočítačem, nebo takový modul navrhněte.
3. Navrhněte a naprogramujte potřebné programové vybavení pro mikropočítač tak, aby ovládal připojený LCD modul a na USB rozhraní simuloval klasické sériové zařízení CDC.
4. Upravte projekt FBTFT tak, aby bylo možno odesílat data pro LCD i přes USB rozhraní.
5. Navržený modul otestujte, vyhodnoťte spolehlivost, rychlost a stabilitu realizovaného modulu.

Seznam doporučené odborné literatury:

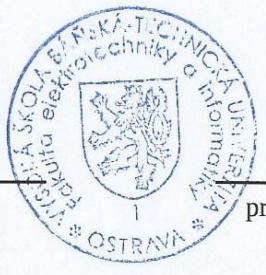
[1] Projekt FBTFT: https://github.com/notro/fbtft/wiki
[2] Knihovna UTFT: http://henningkarlsen.com/electronics/library.php?id=52
[3] Příklad vhodného LCD modulu: STM32F429I DISCOVERY

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Petr Olivka, Ph.D.**

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 .....

Chtěl bych poděkovat panu Ing. Petru Olivkovi, Ph.D. za odborné vedení práce, cenné rady, věcné připomínky a vstřícnost při konzultacích, které mi pomohly tuto práci vypracovat.

Abstrakt

Diplomová práce pojednává o tvorbě malého, přenosného LCD displeje pro operační systém Linux, připojitelná přes USB rozhraní. Jádrem řešení je technologie DisplayLink, která má za majoritní cílovou skupinu počítače se systémem Microsoft Windows. LCD displej je realizován vývojovým modulem STM32F429 DISCOVERY od výrobce ST Microelectronics. Na tomto vývojovém modulu jsou data přijatá přes USB z ovladače udlfb (open-source ovladač omezeně podporující DisplayLink na OS Linux) rozpoznávána, transformována a zobrazována na integrovaném LCD displeji. Práce v úvodu rozebírá analýzu možných řešení a zhodnocuje jednotlivé navržené realizovatelné verze. V závěrečných kapitolách jsou nastíněny možnosti dalšího rozšíření a vylepšení. Zmíněn je také přínos a využití vzniklého řešení v praxi.

Klíčová slova: Raspberry Pi, STMicroelectronics, ST32MF429 Discovery, LCD displej, C, DisplayLink, CDC, USB, udlfb, open-source, embedded programování.

Abstract

This diploma thesis deals with the design of small, portable LCD display for Linux operating system, which can be connected through USB interface. The core of the solution consist of DisplayLink technology, which majority end-point group are computers with Microsoft Windows. LCD display is realized by a development kit STM32F429 DISCOVERY produced by ST Microelectronics. On this development kit the data are received by USB from udlfb driver (open-source driver of some functionality of DisplayLink technology for Linux) are parsed, resampled and finaly displayed on an integrated LCD display. In the main sections the thesis analyses available solutions for the problem and makes the comparison of them. The thesis also discusses the subsequent expansion options and improvements. Also the usage of the product of this thesis is discussed.

Key Words: Raspberry Pi, STMicroelectronics, ST32MF429 Discovery, LCD display, C, DisplayLink, CDC, USB, udlfb, open-source, embedded programing.

Obsah

Seznam použitých zkratek a symbolů

ARM	–	Acorn RISC Machine
BYOD	–	Bring Your Own Device
CDC	–	Communications Device Class
CPU	–	Central Processing Unit
CRC	–	Cyclic Redundancy Check
CSI	–	Camera Serial Interface
CSW	–	Command Status Wrapper
DPS	–	Deska Plošných Spojů
DVI	–	Digital Visual Interface
EDID	–	Extended Display Identification Data
EPA	–	Environmental Protection Agency
FB	–	Framebuffer
FPS	–	Frames per second
FS	–	Full Speed
HDMI	–	High–Definition Multimedia Interface
HS	–	High Speed
HW	–	Hardware
GPU	–	Graphic Processing Unit
GPIO	–	General Purpose Input / Output
I/O	–	Input / Output
IOT	–	Internet of Things
LCD	–	Liquid Crystal Display
LED	–	Light-Emitting Diode
LFSR	–	Linear Feedback Shift Register
OS	–	Operační systém
OTG	–	On–The–Go
QVGA	–	Quarter Video Graphics Array
RAM	–	Random–Access Memory
SD	–	Secure Digital
SoC	–	System on Chip
SPI	–	Serial Peripheral Interface
SSH	–	Secure Shell
SW	–	Software
TFT	–	Thin–Film Transistor
USB	–	Universal Serial Bus
VGA	–	Video Graphics Array

VM – Virtual Machine

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Se stále se rozvíjejícím odvětvím nazývaným jako IoT (Internet věcí) roste také počet různých vestavěných zařízení, které jsou buď jednoúčelové, nebo právě naopak zastávají hned několik rolí ať už podobných či rozdílných. Uživatelé především z oboru informačních technologií, ale i mnozí další, si tak často tato zařízení (jako jsou například NAS servery, multimediální centra, různé automatizační systémy ale i například 3D tiskárny) realizují svépomocí, aby splňovala jejich konkrétní požadavky. Velice populárním základem pro vytvoření některého výše zmíněného cíle bývají malé počítače typu SoC.

Za přijatelnou cenu uživatel získá počítač o velikosti kreditní karty, s dostatečným výkonem, nízkým příkonem a obvykle bez nutnosti aktivního chlazení. Tyto minipočítače nejsou většinou nijak fixované na systémovou platformu a tak v jejich útrobách často nalezneme nejrůznější verze operačních systémů od Androidu přes různé odnože Linuxu a speciální verze Microsoft Windows až po specifické operační systémy daného výrobce [3] [4]. Právě systém Linux je zde zastoupen s majoritním podílem.

Minipočítače, které jsou zde zmiňovány, na úkor svých rozměrů nemají žádná zobrazovací zařízení, která by utvářela vizuální interakci s uživatelem. Obvykle disponují několika typy rozhraní, ke kterým je možno připojit zobrazovací zařízení. To ale nemusí být vždy pohodlné a požadované. Mnohdy uživatel jen potřebuje mít kontrolu nad nějakým procesem či jen provést konfiguraci.

To dalo podnět vytvořit jednoduchý, levný, malý a snadno přenosný displej, který by se dal použít, jak pro konfiguraci či experimentování, tak i pro běžnou práci se zařízením.

Tato diplomová práce je zaměřená na malý grafický LCD displej, který lze připojit přes rozhraní USB k jakémukoliv počítači se systémem Linux. Primárně je však vývoj směřován pro systém OS Linux.

Tento displej je tak možno využít jako mobilní periferii sloužící například pro konfiguraci malých přenosných počítačů, jejichž zástupcem je v této diplomové práci minipočítač Raspberry Pi. Displej pak může být zakomponován do různých technologických řešení, kde nachází tyto minipočítače využití.

V úvodních kapitolách této práce jsou popsány základní technologie, které mohou být použity k vytvoření takového displeje. Je zde zmíněna knihovna FBTFT, která umožňuje vytvořit malý přenosný LCD displej přes rozhraní SPI. Dále je zde rozebráno několik možných řešení displeje připojitelného přes USB, tedy dalšími způsoby jakými by šlo celý produkt realizovat. Je brán ohled na klady a zápory zmíněných řešení a na základě shrnutí všech poznatků je pak zvolen vhodný způsob řešení.

Následující kapitoly pojednávají o vybraném řešení za pomoci technologie DisplayLink, praktické implementaci a samotném řešení.

Dvojice kapitol následující za popisem a objasnění samotného řešení nastiňuje možná vylepšení výsledného produktu této diplomové práce a možné využití celého řešení v praxi. V první

kapitole z této dvojice je uvedeno pár příkladů, které jsou rozebrány na úroveň stanovení stěžejních bodů pro konkrétní rozšíření. Druhá kapitola se pak zabývá samotnými případy uplatnění výsledku práce v praxi a jeho možnostmi.

Poslední kapitola je věnována otestování a porovnání výsledků s implementací, která používá knihovnu FBTFT a připojení přes SPI. V této kapitole je také zmíněno, jak by se dalo vzniklé řešení rozšířit a vylepšit.

2 Analýza

Kapitola pojednává o průzkumu možností, kterými lze připojit LCD displej k počítači (kromě standardního analogového či digitálního rozhraní VGA/DVI/HDMI). Tato analýza byla provedena před samotným započetím práce z důvodu zjištění možností, kterými je možno řešení realizovat. Jedním z podstatných kroků, které vedly ke zhotovení této části, byla také snaha nezvolit slepou uličku, kde by mohlo dojít k nemožnosti dokončit samotné řešení na základě technických či jiných důvodů, a eliminovat tak případné nepředvídatelné komplikace, které by mohly nastat při vývoji. Také byl kladen důraz na tvorbu takového řešení, které bude vyžadovat jen minimální modifikace do budoucna.

Obzvláště důležité bylo zvážit, zda se vydat cestou vývoje vlastního ovladače či softwaru a ten pak zakomponovat do jádra systému. Bylo by možné spojit se s komunitou, která systém vyvíjí. Je zde však málo času v rámci jedné diplomové práce, že by se podařilo tento ovladač dostat oficiální cestou do nějaké distribuce systému Linux, jelikož to komunita zajisté nemá v plánu, a argumenty, proč by tento software neměl být přijat by jistě zahrnovaly fakt, že LCD displej se k těmto minipočítačům dá připojit za pomoci SPI nebo rozhraní CSI. Taková akce vyžaduje dlouhodobou snahu a velké množství práce. Navíc, jak je zmíněno dále v kapitole 2.3, řešení pro tuto problematiku již z části nabízí zakomponovaný ovladač **udlfb**.

Nabízí se tedy možnost se tomuto kroku vyhnout a jít svou vlastní cestou. To by ovšem zajisté znamenalo vytvořit si vlastní odnož, nebo vývojovou větev verze systému Linux a tu pak nadále udržovat kompatibilní a aktuální. Z hlediska ochrany vlastnictví kódu by zde problém nebyl hlavně díky tomu, že systém Linux je pod licencí open-source, která umožňuje prakticky komukoli prohlížet a upravovat zdrojové kódy. Tento směr se však zdá být poměrně zbytečný a neefektivní z jednoho prostého důvodu. Větev by se musela udržovat stále aktuální a vyžadována by zajisté byla její neustálá inovace. Určitě by se muselo na vývoji této oddělené větve podílet více lidí, ne jenom jedna osoba. Tento směr by s sebou nesl nemalou zodpovědnost a velký závazek do budoucna. Pokud je zde však snaha o to, aby výsledek této práce byl užitečný i do budoucna, ať už jako výchozí bod či jen jako soubor poznatků, je vhodné se zaměřit na to, aby výsledný produkt byl co nejvíce kompatibilní s různými verzemi a distribucemi. Pokud by nebyl, jednoduše by zastarával stejným tempem jakým stárne konkrétní verze kernelu a po několika vydaných verzích by se již stal neatraktivním.

Další věcí, která musel být řádně analyzována před započetím práce byla volba vhodného hardware pro realizaci připojitelného zařízení, zde se však nejedná o tak zásadní krok, který by kriticky naboural vývoj a tak mu byla věnována pozornost až na druhém místě. O výběru vhodného hardware je pojednáváno v kapitole 3.1.

Podkapitoly níže popisují jednotlivá možná řešení, která spadají do obou výše zmíněných skupin, rozepsaná trochu konkrétněji, s přihlédnutím na jejich výhody, nevýhody a omezení. Závěrem této části jsou zjištěné poznatky shrnuty a na jejich základě vytyčeny stěžejní body, které pomohly tuto práci vést vhodným směrem tak, aby i v budoucnu byla využitelná.

2.1 Knihovna FBTFT

FBTFT je knihovna nebo přesněji soubor ovladačů, primárně určen pro minipočítač Raspberry Pi. Umožňuje operačnímu systému Linux zobrazovat obrazová data na malém TFT LCD displeji připojenému přes sběrnici SPI. Jako zdrojová data pro LCD displej využívá Linuxového výstupního zařízení frame-buffer. Připojené zařízení se tak zobrazí v adresáři */dev* jako znakové zařízení s názvem *fbX* a ovladač obstará potřebné zpracování dat pro tento připojený displej.

Podle provedených testů, které jsou uvedeny na [20] je schopna tato knihovna u displeje s rozlišením 480×320 dosahovat obnovovací frekvence i 25 FPS, avšak průměrná hodnota se pohybuje mezi 11 až 17 FPS.

Podobná knihovna je také nabízena pro minipočítače Arduino. Její název je TFTLCD a je určena primárně pro TFT LCD displej Adafruit 2.8". Za použití několika dalších knihoven je možné s tímto řešením vytvořit LCD displej k Arduinu, bez potřeby složité a zdlouhavé implementace. Zajisté by se i tato knihovna mohla stát dobrým zdrojem nejen pro inspiraci a ukázky postupů, ale bylo by možné ji i použít pro řešení.

Výhodou této knihovny je velký počet podporovaných LCD displejů, pro které zde už jsou integrované ovladače. Odpadá tedy nutnost implementovat ovladač pro displej, stačí použít některý z velké škály podporovaných zařízení. Další výhodou je již implementovaná část, která obstarává získávání obrazových dat ze zdroje, čili frame-bufferu, a jejich následné zpracování.

Nevýhodou se jeví nutnost implementace ovladače pro USB komunikaci, který zajišťuje komunikaci se zařízením připojeným přes dostupné USB rozhraní a jeho následné propojení s knihovnou a zavedení do kernelu systému, jak již bylo zmíněno a odůvodněno v úvodu této kapitoly. Právě ono samotné udržení ovladače v kernelu a jeho přenositelnost na další verze kernelu může být kritická. Bylo by zde také nutné důkladně zvážit formu jakou by se data odesílala, zajisté by muselo být využito nějaké vhodné kompresní metody pro redukci přenášených dat.

2.2 Samostatná aplikace

Samostatnou aplikací, která při svém běhu zpracovává data a odesílá je po virtuálním sériovém portu by taktéž mohla být řešením k sestavení požadovaného displeje připojitelného přes rozhraní USB. Taková aplikace by na vstupu přijímala obrazová data odesílaná přes SPI od knihovny FBTFT, nebo rovnou z frame-bufferu (to by však nebylo nutné, jelikož je k dispozici knihovna FBTFT, která tuto část již řeší). Aplikace by dále data zpracovala, zkomprimovala a odeslala přes virtuální sériový port na USB zařízení, které by implementovalo rozhraní CDC (Comunication Device Class). Na tomto zařízení by se data dekomprimovala a zobrazila.

Možným řešením tak je také získávat data ze znakového zařízení *fb0* bez využití FBTFT, ta pak odesílat na zařízení a zde již provádět zpracování, a nebo si data na straně USB Host připravit do požadované podoby a až ta poté odeslat. To by ovšem vyřešilo jen jednoduchou duplikaci již jednoho zobrazovacího zařízení.

2.3 DisplayLink

Do analýzy bylo také zahrnuto řešení nazývané DisplayLink, které za pomocí vlastního USB protokolu a ovladačů, které jsou již implementovány v novějších kernelech systému, řeší problém USB zařízení, které je schopno utvořit zobrazovací jednotku. Na trhu jsou nabízeny různé sériově vyráběné displeje a konvertory připojitelné přes USB, které toto rozhraní implementují. Na internetu jsou k dispozici ovladače pro systémy Microsoft Windows, Linux, Android i Mac OS.

Také byla vytvořena malá komunita uživatelů, která se touto technologií zabývala a snažila se o vytvoření ovladače v době kdy DisplayLink byl podporován jen na systémech Windows (viz článek [10] a webová stránka [11]). Ovladače, které vznikly v této malé komunitě jsou volně ke stažení s otevřeným zdrojovým kódem. Nejstarší ovladač je možné najít na webu autora [11], Floriana Echtlera, a na jeho GitHubu pod názvem **libdlo**, mladší a propracovanější, který je již součástí nových kernelů určitých systémů Linux je nazván **udlfb**. Tyto ovladače ukazují důležité principy a tím se mohou stát dobrou oporou a zdrojem informací při tvorbě zařízení. Florian Echtler také na svém webu zveřejnil řadu poznámek a informací k řešení DisplayLink, na které narazil během své práce.

2.4 Shrnutí

Nejvhodnějším řešením, se po podrobném prozkoumání dostupných možností jeví využít dostupného řešení DisplayLink. Velkým kladem je integrace a podpora nejen UNIXových systémů, ale také podpora pro operační systémy Microsoft Windows, Mac OS X a Android. Výsledkem by tak mohlo být USB zařízení (přesněji řečeno implementovaná knihovna v takovém zařízení), které bude multiplatformní. Knihovna FBTFT by samozřejmě byla využita jako zdroj ovladačů pro další displeje. Pokud by tedy bylo vyvíjeno zařízení například pro kit s mikroprocesorem bez integrovaného LCD displeje, mohly by být využity ovladače z knihovny FBTFT pro LCD displej, který by byl připojen k tomuto mikroprocesoru například za pomoci sběrnice SPI.

Velikou výhodou je, že ovladače pro Linux jsou hotové, otestované a jsou drženy ve vývojové větvi, která je udržována velkou komunitou po celém světě. Je tedy potřeba „pouze“ vyvinout zařízení, které bude zvolené rozhraní podporovat, bude s ním umět komunikovat a bude rozumět příkazům kterými ovladač DisplayLink navenek vystupuje. Data z knihovny DisplayLink jsou také zpravidla šifrovaná a komprimovaná. Bude tedy nutné tyto části implementovat.

Nutností bude analyzovat zdrojové kódy volně dostupných ovladačů pro zjištění algoritmu, jakým jsou data zašifrovaná a zkomprimovaná. Při analýze by značně mohlo pomoci jakékoliv originální zařízení, které DisplayLink podporuje. Bylo by jedno zda se jedná o celý LCD displej nebo adaptér, který disponuje tímto rozhraním. Odposlouchávání jeho komunikace v různých situacích může mnohé napovědět a usnadnit.

Využití knihovny FBTFT a její následné doplnění o USB rozhraní, či využití samostatné aplikace, která obstarává samotnou komunikaci i na nižších vrstvách se při srovnání s řešením DisplayLink nejeví jako nejvhodnější cesta. Vše potřebné pro hostitelskou část je již zakompo-

nováno do každého novějšího operačního systému. Bylo by tedy velmi neefektivní tohoto řešení nevyužít a psát jej celé od začátku. Jedná se o otestované a plně optimalizované řešení s dlouhodobou podporou a integrací na takové úrovni, že jeho zprovoznění s originálními komponenty vyžaduje minimální zásah uživatele.

3 Řešení ve spojení s DisplayLink

V průběhu analýzy možných a realizovatelných řešení bylo navrženo vytvořit na straně displeje jednotku, která se bude po připojení zařízení do USB portu tvářit jako zařízení typu DisplayLink 3.3. Na minipočítači či systému Linux se pak využije podpora výstupu DisplayLink Framebuffer a použije se opensource ovladač libdlo, vydaný společností DisplayLink. Takto vzniklé zařízení by mělo oproti předchozímu návrhu ještě jednu velkou výhodu. Nebylo by závislé na platformě, neboť DisplayLink má pro svá zařízení širokou škálu ovladačů s podporou operačních systémů Microsoft Windows, Mac OS X, Android a Ubuntu.

3.1 Hardware

K řešení daného problému byl použit hardware, který je popsán v podkapitolách níže. Hovoříme zde hlavně o vývojovém kitu 32F429IDISCOVERY od STMicroelectronics, který představuje samostatnou jednotku s LCD displejem, která je připojitelná přes USB rozhraní k počítači.

Dále jsou uvedeny informace o minipočítači Raspberry Pi, ke kterému je možno přes USB připojit LCD displej a plně tak nahradit HDMI nebo analogový výstup pro připojení standardního monitoru. Hlavní myšlenka využití displeje přes USB rozhraní je především možnost konfigurace minipočítače bez nutnosti připojovat velký monitor jako periferii či připojovat se vzdáleně přes SSH. Raspberry Pi bylo proto vybráno za zástupce široké řady minipočítačů jelikož je určen pro open–source platformu, je kolem něj vytvořena rozsáhlá komunita uživatelů a vývojářů, kteří vytváří nepřebernou řadu projektů a experimentů. V neposlední řadě se Raspberry Pi těší čím dám větší oblibě mezi uživateli.

V poslední podkapitole je popsán běžně dostupný LCD displej Lenovo ThinkVision LT1421, který lze připojit přes USB a využívá rozhraní DisplayLink pro přenos obrazových dat.

3.1.1 Vývojový kit s LCD displejem

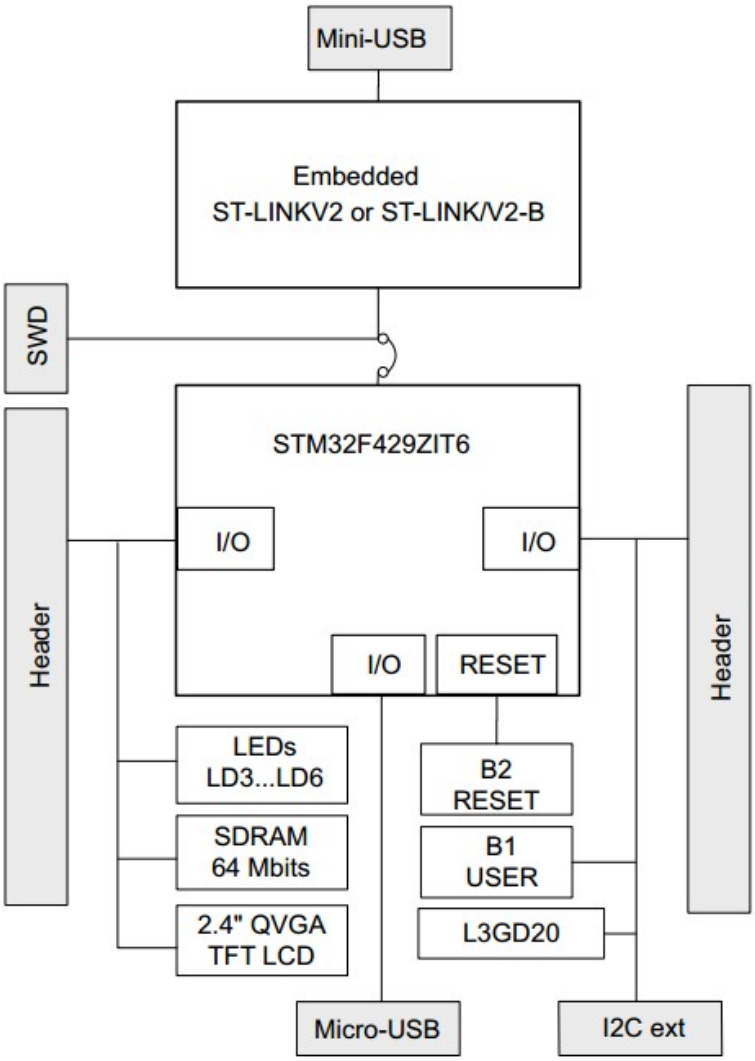
Značná pozornost byla věnována také fyzickému (hardwarovému) řešení ovládání LCD displeje a také samotnému LCD displeji. Nabízely se tyto možnosti: Buď využít samostatný LCD displej, navrhnout a vytvořit desku plošných spojů, osadit ji vhodnými komponenty a na tomto modulu dále pokračovat ve vývoji nebo využít nabídky trhu, kde je dostupné velké množství sériově vyráběných modulů, které již mají integrovaný displej nebo jej dovolují snadno připojit. Na trhu je dostupná velká škála tzv. kitů, od jednoduchých DPS s osazeným mikrokontrolérem a vývody, přes sofistikovanější např. s připravenými komponenty jako jsou LED diody či tlačítka až po hotové DPS s LCD displeji, USB rozhraním nebo množstvím I/O pinů. abychom zaručili snadnou reprodukovatelnost, byla zvolena druhá možnost, konkrétně tedy využít některého hardwarového řešení, které je běžně k sehnání, oproti vývoji vlastního hardware, který by zajisté nebyl na takové úrovni jako výrobky které jsou k mání.

Obrázek 1: STMicroelectronics - 32F429IDISCOVERY [2]

Vybrán byl vývojový kit 32F429IDISCOVERY od STMicroelectronics, který je osazen mikroprocesorem STM32F429 architektury ARM z rodiny mainstreamových procesorů Cortex–M4. Tato řada procesorů nachází široké uplatnění v embedded systémech. Procesor je taktován na frekvenci 180 MHz. Modul disponuje především 2.4" TFT LCD Displejem, 2 MB Flash paměti, 256 KB RAM, 64 Mbit SDRAM, nebo třeba dvěma USB porty (USB OTG FS a USB OTG FS/HS). TFT LCD displej má rozlišení 240×320 pixelů a 262 tisíc barev na pixel. Dále je na modulu umístěn také mikroprocesor s debuggerem ST-LINK, takže umožňuje připojení k počítači v debug modu, což zajisté usnadňuje vývoj.

Pro přehlednost a ucelení náhledu na celý vývojový kit je na obrázku 2 blokově znázorněno propojení jednotlivých periferií, celkový pohled na DPS vývojového kitu je pak na obrázku 1. Na blokovém schématu je uprostřed umístěn mikroprocesor s označením STM32F429ZIT6 na jeho vstupně/výstupní piny jsou připojena dvě USB rozhraní, z nichž jedno využívá integrovaného debuggeru ST–LINK V2 nebo ST–LINK/V2–B. Dále jsou patrné periferie jako LED diody LD3 – LD6, SDRAM, QVGA TFT LCD displej, I2C sběrnice, tlačítka B1 a B2. Zajímavou součástkou, kterou tato zařízení běžně nedisponují je také gyroskop ST MEMS L3GD20.

STMicroelectronics ke svým vývojovým kitům poskytuje dokumentaci a programátorskou podporu ve formě ovladačů, demo programů, odkazů na různé frameworky (např. .NET Micro



Obrázek 2: STMicroelectronics - 32F429IDISCOVERY - blokové schéma periferií [2]

Framework) nebo také doporučení na vývojová prostředí a nástroje, ve kterých lze programové vybavení vyvíjet tak aby bylo s daným kitem kompatibilní. Podrobné technické parametry a specifikace, dokumentace a odkazy na stažení software, toolchainů, demo programů a frameworků jsou dostupné na [2]

3.1.2 Raspberry Pi

Raspberry Pi 3 je jednodeskový minipočítač o velikosti kreditní karty, vyvíjený britskou nadací Raspberry Pi Foundation.

Pro naše účely byla použita verze 2 model B. Ta je osazena čtyřjádrovým procesorem ARM Cortex–A7 o frekvenci 900 MHz a 1 GB RAM, který je označován za jeden z nejméně energeticky náročných procesorů rodiny ARM. Na rozdíl od předchozí verze minipočítače má Raspberry Pi 2

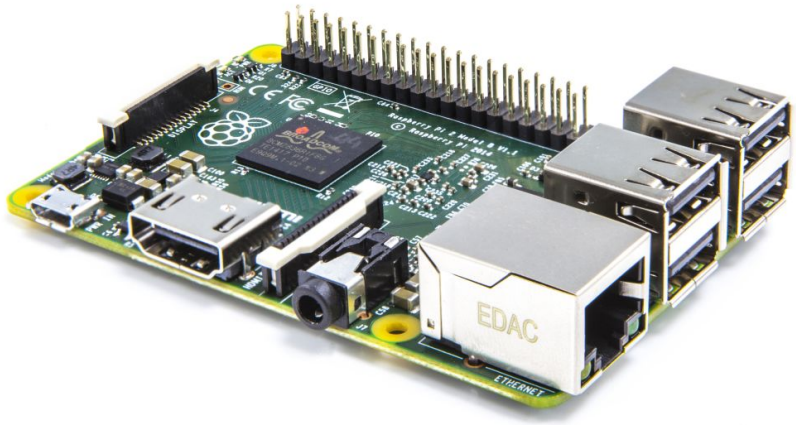
celkem čtyři USB porty, takže již není potřeba používat aktivní USB hub, jako tomu bylo u staršího modelu při použití více než dvou periferních zařízení. Minipočítač obsahuje také grafické jádro VideoCore IV 3D s výstupem přes HDMI výstupem, slot na microSD paměťovou kartu, rozhraní pro připojení kamery CSI, obecné rozhraní SPI, Ethernetový port 10/100 Mbits a konektor typu Jack 3.5 mm pro výstup audia anebo nově i jako kompozitní video výstup.

Slot pro SD kartu je u tohoto minipočítače využit jako disk pro instalaci systému.

Operační systém pro Raspberry Pi tvoří převážně verze Linuxu vycházející z distribucí Debian a Arch Linux. Dostupné jsou tyto systémy: Raspbian, Fedora 21, Ubuntu 15.04 Mate, RiscOS, Gentoo, Pinet a OSMC. Novinkou v kompatibilních operačních systémech je produkt společnosti Microsoft. Od ní je dostupný systém Windows 10 IOT Core, který je určen převážně pro využití v oblasti Internet of Things.

Raspberry Pi je možno také rozšířit o další zařízení, která mohou představovat desky s obvody, připojitelné za pomocí rozhraní GPIO. Tato rozšíření pak přidávají možnosti Raspberry Pi, které samotný minipočítač postrádá.

Podrobné technické specifikace minipočítače Raspberry Pi lze najít na [1].



Obrázek 3: Raspberry Pi 2 model B

3.1.3 Lenovo ThinkVision LT1421

Jako testovací zařízení byl během práce využit monitor ThinkVision LT1421 (vyobrazen na 4), jehož výrobcem je výrobce počítačů Lenovo. Jedná se o jeden z dobře dostupných monitorů ve své kategorii.

Obrazovka s uhlopříčkou čtrnáct palců a poměrem stran 16:9 podporuje rozlišení 1366×768 pixelů, které je nejčastěji využívaným rozlišením na běžně prodávaných noteboocích. Při váze 0.84 kg a rozměrech $335 \times 217.9 \times 21.5$ milimetrů se jedná o relativně skladnou a snadno přenosnou periferii.

Tabulka 1: Spotřeba displeje Lenovo ThinkVision LT1421 [9]

	Spotřeba [W]
Maximální	**5**
Běžné	**4.2 (EPA 5.1 st.)**
Stand-by	**<0.1**
Uspaný	**<0.1**
Vypnutý	**0**

Napájení zařízení je realizováno přes USB port. V balení je dodáván rozdvojený USB kabel (často označován jako typ Y), který je používán např. u přenosných pevných disků či dalších periferií vyžadující zdroj s možností odběru vyšší proudové zátěže. V případě potřeby (převážně u malých zařízení jako jsou Raspberry Pi, mobilní telefon, tablet a dalších) je však možné jeden USB vstup zapojit do zařízení a druhý vstup do napájecího adaptéru nebo do druhého počítače, a tím poskytnout LCD displeji dostatečně silný zdroj napájení (viz výrobcem udávané hodnoty spotřeby v tabulce 1).

Obrázek 4: Lenovo ThinkVision LT1421

3.2 Software

Podkapitola popisuje konkrétní softwarové standardy a způsoby implementace, které se používají v praxi a byly pro realizaci LCD displeje využity. V první řadě je zmíněna třída Communication Device Class, která je využita při komunikaci za použití USB protokolu. Dále je detailněji popsána technologie DisplayLink, umožňující připojit k počítači displej nebo konvertor přes rozhraní USB, Ethernet, nebo Wi–Fi.

3.2.1 Communication Device Class

Způsob komunikace za použití CDC (Communication Device Class) je jeden z nejjednodušších a nejrozšířenějších způsobů, jak přenášet data mezi počítačem (USB Host) a mikroprocesorem. Jedná se o virtuální sériový port. Za pomoci vhodného ovladače a patřičného programového vybavení na straně mikroprocesoru vytvoříme simulovaný sériový port, který však využívá namísto sériové linky USB. Často je využíván k připojení sériových zařízení k počítači, ať už z důvodu absence sériových portů, vzdálenosti zařízení od počítače nebo kvůli bezpečnostním opatřením.

Nevýhodou by se mohla jevit rychlost komunikace tzv. baudrate, která se využívá u sériových portů jako jeden ze vstupních parametrů rozhodujících o tom, jakou přenosovou rychlostí bude komunikováno. Zde však touto rychlostí omezeni nejsme, jelikož komunikujeme po jiném médiu a hlavně na odlišném protokolu než využívají standardní sériové porty. Teoreticky je možné dosáhnout rychlosti komunikace přes USB 2.0, tj. 60 MB/s. Tato hodnota je však opravdu pouze teoretická a v praxi zpravidla není dosahováno takových výsledků, jelikož přenos reálných informací zahrnuje kromě „užitečných“ dat i další informace jako jsou adresy, CRC, řídicí tokeny atd. Testy ukazují, že rychlost při využití virtuálního sériového portu se pohybuje kolem 1.5 MB/s.

V případě využití CDC v komunikaci pro přenos obrazu se však stává „hrdlem láhve“ právě rychlost operace s buffery na obou stranách. Záleží tedy na algoritmech, které zpracovávají data přijatá do bufferů, dále s nimi operují a po zpracování buffer vyprázdní. Potvrzení po přijetí je vyžadováno vždy a zpravidla jde ruku v ruce společně s nějakým příkazem indikujícím, že cílový buffer je vyprázdněn a může být započato odesílání dalších dat.

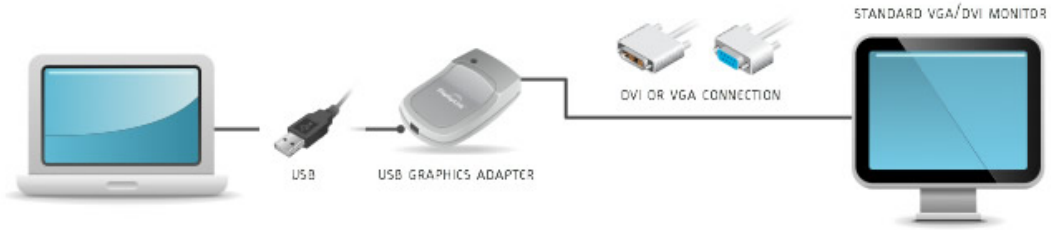
3.3 DisplayLink

Grafická technologie USB DisplayLink je kombinace hardwarové a softwarové technologie, která umožňuje připojit takřka jakýkoli druh monitoru k počítači přes rozhraní USB, Ethernet či Wi–Fi. Jedná se o technologii, již podporuje řada průmyslově vyráběných zařízení (na trhu běžně dostupných), využívaných především k řešení problému, který je často nazýván jako BYOD (Bring Your Own Device). Technologie tak dává uživateli nezávislost na platformě při potřebě zobrazení obrazového výstupu u ultrabooků, tabletů a také MacBooků. DisplayLink k těmto zařízením umožňuje připojit více monitorů a zvýšit tak jak uživatelovo pohodlí tak i produktivitu. Mezi udávané výhody se řadí úspora pořizovacích nákladů při případném upgradu hardwaru

a především až 80% energetická úspora oproti běžnému řešení zapojení většího množství monitorů za použití adekvátního počtu grafických karet. [6]

Produkty využívající USB technologii DisplayLink mohou být užívány prakticky s jakýmkoliv počítačem, který disponuje USB portem. Je jedno, zda se jedná o starší USB Standard–A port a nebo nový USB–C standard. [6]

Potencionálním problémem je napájení takového zařízení. Zapotřebí je dostatečně silný zdroj (USB port) nebo případně může být využita powerbanka při připojení k mobilnímu zařízení.



Obrázek 5: Schéma standardního zapojení DisplayLink. Zdroj: www.displaylink.com

3.3.1 Technologie DisplayLink

Technologie je navržena pro co nejjednodušší přidání doplňujících monitorů k počítači přes USB. Za pomoci DisplayLink je možno připojit až 6 displejů bez nutnosti přidání další grafické karty. Uživatel jednoduše nainstaluje potřebný ovladač a software podpory DisplayLink na počítač, připojí nový monitor do USB, pokud se jedná o zařízení DisplayLink s USB rozhraním, nebo využije adaptéru či dokovací stanice, do které je zapojen monitor standardním kabelem VGA-/DVI/HDMI. Schéma takového zapojení je znázorněno na obrázku 5.

Tato technologie využívá volných prostředků CPU a GPU ke zpracování informací pro DisplayLink hardware, který je připojen do USB. Za pomoci různých algoritmů rozhodne o tom, která část obrazovky potřebuje být obnovena. Neobnovuje se pokaždé celá obrazovka, ale pouze její části. Obrazový signál tedy nepřichází standardně z počítače způsobem, jaký je běžný při klasickém připojení monitoru. Zpracování informací se odehrává mezi softwarem na počítači a hardwarem DisplayLink na straně monitoru. Obnovování, komunikace a komprese je zajištěna kompresní technologií DL2+ nebo DL3. Tyto adaptivní kompresní technologie automaticky vyvažují kompresní metody na základě obsahu, dostupného výkonu CPU a propustnosti USB tak, aby bylo dosaženo co nejlepší kvality obrazu v daný moment. Zkomprimovaná data jsou odeslána přes standard USB 2.0, nejvyšší možnou rychlostí. Následně pak čip se programovým vybavením DisplayLink, obsažený v monitoru, dekóduje komprimovaná data zpět na obrazová data a zajistí jejich správné zobrazení. [5]

4 Implementace

Tato kapitola je věnována stěžejní části práce. Tou je samotná implementace a konfigurace, jak na straně minipočítače Raspberry Pi, tak na straně vývojového kitu. Na straně Raspberry Pi se jedná především o sestavení jádra operačního systému a o konfiguraci software dodávaného společností DisplayLink a jeho zavedení do systému. Sestavení jádra (kernelu) se však týká pouze starších systémů. Novější již v sobě mají integrovány potřebné ovladače ve výchozím nastavení. Na straně 32F429IDISCOVERY implementace zahrnuje část vývoje na nízké úrovni (enumerace, inicializační sekvence zařízení a úprava vhodného protokolu pro komunikaci) a také část zpracování a zobrazení přijatých obrazových dat na integrovaném TFT LCD displeji.

K vývoji softwarového vybavení pro kit 32F429IDISCOVERY bylo jako vývojové prostředí použito Atollic TrueSTUDIO for ARM v5.4.0 Lite, které je dosti podobné rozšířenému a hojně využívanému vývojovému prostředí Eclipse, které se využívá pro vývoj v jazycích Java, C/C++, PHP, a dalších.

V následujících podkapitolách jsou rozepsány kroky, které bylo potřeba provést, aby vývojový kit STM32F429 Discovery emuloval displej s rozhraním DisplayLink, USB Host jej takto rozpoznal a aby kit byl schopen přijatá data zpracovat, případně dešifrovat, dekomprimovat a zobrazit na svém integrovaném LCD displeji o rozlišení 240×320 pixelů. Podkapitoly jsou řazeny od prvotního připojení zařízení až po zobrazení dat na displeji.

4.1 Analýza a konfigurace zařízení DisplayLink

Pro lepší pochopení, jak zařízení DisplayLink funguje a komunikuje byl použit displej Lenovo ThinkVision LT1421, který je popsán v kapitole 3.1.3. Za pomoci softwarových nástrojů Thesyncon USB Descriptor Dumper[13] a USBlyzer[12] byla monitorována komunikace a přenášená data. Později při vývoji a ladění enumerace byl využit program Wireshark[14], který je primárně určen pro monitorování síťového provozu. Po doinstalování USBPcap[15] je však schopen zachytávat i provoz po USB sběrnici a na rozdíl od programu USBlyzer zobrazuje i podrobná surová data, a tak jsou exportované výpisy z něj detailnější. USBlyzer převyšuje USBPcap v přehlednosti komunikace, avšak není schopen zobrazit surová data jednotlivých částí hlavičky paketu jen jeho datové části. Ideálního stavu bylo dosaženo při kombinaci těchto dvou nástrojů.

Reverse-Engineeringem DisplayLink produktů se zabývali Florian Echtler a Chris Hodges, kteří své výsledky prezentovali na konferenci 26th Chaos Communication Congress a na webu zveřejnili o své práci článek[10]. Florian Echtler pak na svém osobním webu[11] prezentoval pár důležitých postřehů ze své práce spolu s odkazy na software „tubecable“, který vyvinul výhradně pro odposlech komunikace protokolu DisplayLink.

Práce autorů se zakládala na tom, že v době, kdy tento článek publikovali, nebyl k dispozici ovladač zařízení DisplayLink pro systémy Linux. Proto se rozhodli za pomoci zkoumání kódu ovladače po dekompilaci a komunikačního provozu DisplayLink zjistit dostatek informací, aby byli schopni si vytvořit vlastní ovladač pro tato zařízení, který by je dovoloval užívat na operač-

ním systému Linux. Na základě jejich aktivity pak společnost DisplayLink zveřejnila open-source ovladač **libdlo**, který však neobsahoval inicializační sekvence a nepoužíval kompresi, takže de facto odhalili jen část protokolu, který zajišťuje komunikaci.

Florian Echtler na svém webu [11] popisuje i konkrétní příkazy, které jsou používány při komunikaci, dále zveřejňuje jaká komprese je pravděpodobně použita a jak jsou obrazová data šifrována.

Poznatky obou autorů byly pro vývoj výsledného LCD displeje velmi hodnotné.

4.2 Strana USB Host

Nejprve byla věnována pozornost konfiguraci na straně počítače, tedy USB Hosta. Priorita této volby byla zvolena na základě potřeby zprovoznit originální LCD displej Lenovo LT1421 na základě odchytávání paketů pak postupovat ve vývoji druhé strany (USB Client). Podkapitola shrnuje poznatky, které byly zjištěny během konfigurace a testování na různých systémech a platformách.

4.2.1 Raspbian a DisplayLink

Aby mohlo zařízení typu DisplayLink s počítačem, na kterém běží systém Linux, spolupracovat, bylo nutné provést konfiguraci samotného systému. Toto nastavení je vhodné provést kompilací jádra systému Linux s podporou Displaylink USB Framebuffer. Takto sestavený systém (Raspbian Wheezy) se v našem případě nahrál na SD kartu a vložil do MicroSD slotu na minipočítači Raspberry Pi. Následně bylo zapotřebí vytvořit soubor xorg.conf s patřičnou konfigurací.

U novějších verzí jader je již ovladač **udlfb** zakomponován v kernelu. Proto není nutno nic instalovat, ani sestavovat vlastní kernel. Po připojení zařízení DisplayLink do USB portu se zařízení správně zaregistruje (lze ověřit pomocí příkazu **lsusb**) a dále se sám načte modul **udlfb** (opět lze ověřit příkazem **lsmod**). Ve složce **dev** přibude nové zařízení **fb1**. Dále je zapotřebí vytvořit konfigurační soubor v /etc/X11/xorg.conf, který specifikuje nastavení obrazových vlastností, výstupních obrazových zařízení a uspořádání monitorů. Obsah souboru je zobrazen ve výpisu 1.

Pokud by uživatel požadoval více připojených monitorů, je zapotřebí duplikovat sekce **Device**, **Monitor** a **Screen**, aby jejich počet byl roven počtu připojených monitorů. V takto duplikovaných sekcích je nutné upravit identifikátory a namapovat sekci **Device** na správné zařízení **fb**. Sekce **ServerLayout** slouží ke specifikaci, jak budou jednotlivé obrazovky uspořádány. U položky **Screen** následuje číselný identifikátor obrazovky, unikátní identifikátor, který odkazuje na danou sekci Screen, poloha (reprezentovaná hodnotami "**RightOf**"– napravo a "**LeftOf**"– nalevo) a identifikátor obrazovky, vůči které je poloha myšlena.

Po konfiguraci je nutné systém restartovat s již připojeným LCD displejem. Po nabootování do systému se zpravidla na připojeném USB displeji zobrazí bílé a růžové pruhy, které cca po 5 – 10 vteřinách zmizí a objeví se obraz.

```
Section "Device"
  Identifier "DisplayLinkDevice"
  driver "fbdev"
  Option "fbdev" "/dev/fb1"
  Option "ShadowFB" "off"
EndSection

Section "Monitor"
  Identifier "monitor1"
EndSection

Section "Screen"
  Identifier "screen1"
  Device "DisplayLinkDevice"
  Monitor "monitor1"
EndSection

Section "ServerLayout"
  Identifier "default"
  Screen 0 "screen1" 0 0
EndSection
```

Výpis 1: Struktura konfiguračního souboru /etc/X11/xorg.conf

4.2.2 Ubuntu a DisplayLink

Stejně jako u Raspbianu a ostatních systémů s novými jádry pro Raspbery Pi a další minipočítače (např. Armbian), jak bylo popsáno v kapitole 4.2.1, má i Ubuntu ve svém jádru integrováno velké množství ovladačů a modulů. Mezi nimi je taktéž **udlfb**.

Postup připojení a zprovoznění DisplayLink zařízení je na odnoži Ubuntu totožný s postupem zprovoznění na systému Raspbian, který je popsán v kapitole 4.2.1.

U novějších zařízení DisplayLink je možno použít ovladač z webových stránek výrobce DisplayLink a postupovat podle návodu na **??**. Tento ovladač však podporuje pouze zařízení připojitelná přes rozhraní USB 3.0.

Během testování však bylo zjištěno, že připojení zařízení DisplayLink nemusí fungovat správně na virtualizovaném systému. Konkrétně se tato nekompatibilita objevila za použití softwaru VMware 12.0.0 pro virtualizaci systémů Ubuntu 64-bit 14.04 (Trusty Tahr), Ubuntu 32-bit 14.04

(Trusty Tahr), Ubuntu 64-bit 14.10 (Utopic Unicorn), Ubuntu 32-bit 14.10 (Utopic Unicorn), Ubuntu 64-bit 15.04 (Vivid Vervet) Ubuntu 32-bit 15.04 (Vivid Vervet).

Obdobné problémy se stejnými virtualizovanými operačními systémy byly zaznamenány i na software Oracle VM VirtualBox v nejnovější verzi 5.0.14 r105127. Nutno dodat, že pro připojení zařízení DisplayLink, bylo do tohoto emulátoru nutno doinstalovat doplněk Oracle VM VirtualBox Extension Pack verze 5.0.14 r105127 pro podporu standardu USB 2.0.

4.3 Strana USB Client

Po úspěšném zprovoznění a konfiguraci USB Hosta přišla řada na druhou část, kterou tvoří USB Client. Tato strana vyžadovala řádově mnohem více implementace a různého nastavení. Šlo především o správnou konfiguraci vývojového modulu, nastavení vhodných provozních parametrů, zjištění a vývoj deskriptoru USB zařízení, analýzu a vývoj vhodného komunikačního protokolu pro přenos dat, tak aby byl co nejpodobnější protokolu, který využívá technologie DisplayLink. Velkou část implementace také zahrnuje zpracování a zobrazování obrazových dat, přijatých na zařízení. Velkou pozornost si vyžádala také analýza ovladače **udlfb** pro Linux, který na tomto systému obstarává veškeré procesy týkající se USB LCD displejů. Dále také tato část obnášela velké množství řešení různých problémů, které nastaly během vývoje. Podkapitoly níže tyto jednotlivé body postupně rozvádějí a objasňují jejich funkci, princip a řešení.

4.3.1 Deskriptor zařízení DisplayLink

Při USB komunikaci hraje důležitou roli tzv. deskriptor, což je konfigurace, která se načte ze zařízení do počítače při připojení zařízení do USB portu. Provede se něco na způsob handshake a zařízení o sobě počítači pomocí deskriptoru předá informace o tom, co je zač, jak s ním má počítač komunikovat, kdo je jeho výrobce, jaký ovladač se má použít, jakou rychlost vyžaduje USB zařízení pro komunikaci, jestli potřebuje napájení, nebo je napájeno externě, atp.

Tento deskriptor je pro každé zařízení specifický. Pro open-source projekty není problém na Internetu kódy s touto konfigurací najít, avšak pokud se jedná o zařízení, které je chráněno nějakou licencí a ve své podstatě nemá žádný zveřejněný kód, je tento deskriptor poměrně náročné získat nějakou úvahou či reengineeringem.

Na druhou stranu tento problém již řešil nejeden vývojář a tak společnost Thesyncon Software Solutions GMbH & Co. KG poskytuje freeware program s názvem Thesyncon USB Descriptor Dumper [13], který po připojení zařízení daného typu zobrazí jeho deskriptor. V rámci ověření a abychom předešli pozdějším nesrovnalostem, byl tento software otestován na projektu vygenerovaném z STM32Cube MX s komunikací CDC přes USB. Tento projekt byl zvolen z toho důvodu, že byly k dispozici všechny jeho zdrojové kódy z předcházející analýzy a pokusů, které byly ověřené a funkční. Deskriptor po připojení zařízení byl totožný s tím, který byl ve zdrojových kódech nakonfigurován.

Následně po otestování byl obdobným způsobem diagnostikován sériově vyráběný LCD displej Lenovo LT1421 s rozhraním DisplayLink. Po připojení do počítače a za pomoci programu Thesyncon USB Descriptor Dumper byl zjištěn deskriptor pro toto zařízení, který byl využit pro naše zařízení emulované na kitu STM32F429 Discovery.

4.3.2 Typ USB komunikace

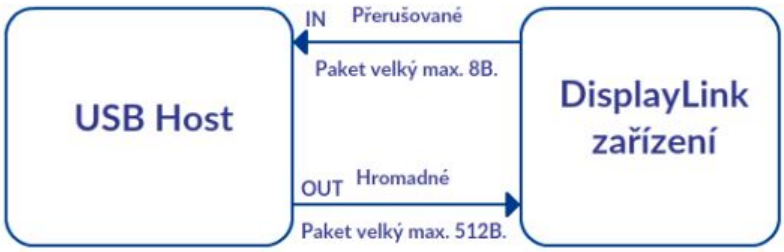
Před započetím vývoje bylo nutno zvážit a zvolit, jaký typ přenosu dat, který je využíván protokolem USB ke komunikaci, bude pro výsledné zařízení vhodný. Bylo zapotřebí udělat analýzu a zjistit, jaký typ je vlastně využit při přenosu se standardními zařízeními DisplayLink. Na výběr se nabízí tyto čtyři možnosti:

Řídicí přenosy jsou použity k zajištění přenosu konfiguračních / řídicích / stavových dat mezi klientským SW a zařízením [7].

Izochronní přenosy jsou přenosy konstantní frekvencí s tolerovanými chybami. Mají garantovaný přístup k USB pásmu s omezeným zpožděním, garantovaný konstantní tok dat přes „rouru“ (pipe) po celou dobu, kdy jsou přenášena data. V případě chybného doručení dat v důsledku chyby nejsou data vysílána znovu [7].

Přerušované přenosy slouží zařízením, která potřebují poslat nebo přijmout data relativně zřídka, ovšem požadují rychlé doručení. Poskytují garantovanou maximální periodu, ve které je požadavek obsloužen a opětovné vyslání v případě chyby při doručení [7].

Hromadné přenosy jsou navrženy pro zařízení, která potřebují přenášet velké bloky dat v různých chvílích, kdy může přenos použít veškeré dostupné přenosové pásmo. Nabízí přístup k USB na základě aktuálně dostupné přenosové kapacity, opakování přenosů v případě chyb na sběrnici, garanci správného doručení dat [7].



Obrázek 6: USB komunikace zařízení DisplayLink

Analýza komunikace s displejem Lenovo LT1421 a prozkoumání použité technologie napovědělo, že se jedná o hromadné a přerušované datové přenosy. Zařízení DisplayLink využívá dva

z šestnácti vstupních a šestnácti výstupních dostupných endpointů. Jeden ve vstupním (IN) a druhý ve výstupním (OUT) režimu. Výstupní využívá hromadných přenosů a vstupní přerušovaných. Tyto údaje jsou patrné z odposlechnutého deskriptoru 2. Celou situaci znázorňuje blokový diagram 6.

```
Endpoint Descriptor:
------------------------------
0x07  bLength
0x05  bDescriptorType
0x01  bEndpointAddress  (OUT Endpoint)
0x02  bmAttributes  (Transfer: Bulk / Synch: None / Usage: Data)
0x0200  wMaxPacketSize  (512 Bytes)
0x00  bInterval

Endpoint Descriptor:
------------------------------
0x07  bLength
0x05  bDescriptorType
0x82  bEndpointAddress  (IN Endpoint)
0x03  bmAttributes  (Transfer: Interrupt / Synch: None / Usage: Data)
0x0008  wMaxPacketSize  (8 Bytes)
0x04  bInterval
```

Výpis 2: Decriptory endpointů displeje Lenovo LT1421

STM32CubeMX nabízí pro prvotní konfiguraci projektu sadu tříd, které korespondují se standardy používanými při vývoji různých USB zařízení. Tyto třídy implementují transportní protokoly, sady příkazů, deskriptory zařízení, konfigurační deskriptory, rozhraní atd. Je tak možno jednoduše nakonfigurovat a vygenerovat třídy, na které se můžeme odkazovat ve vyvíjeném projektu a využívat jejich funkcionalitu v našem kódu. Podrobný popis jednotlivých tříd je možno nalézt v dokumentu na webu STMicroelectronics [8]. Následuje zkrácený výpis klíčových rozdílů, na základě kterých byla zvolena vhodná třída pro započetí vývoje LCD Displeje využívajícího komunikaci za použití řešení DisplayLink.

Human Interface Devices (HID) třída, která komunikaci realizuje se dvěma endpointy. Vstupní endpoint využívá přerušovaných přenosů, výstupní endpoint může komunikovat za pomocí přerušovaných nebo řídicích příkazů.

Mass storage class (MSC) komunikuje za použití protokolu Bulk–Only Transport (BOT). Tento protokol komunikuje za využití jednoho nulového endpointu a na základě přijatého CSW bloku jej nastaví jako vstupní, výstupní nebo nulový s použitím hromadných přenosů.

Device firmware upgrade (DFU) využívá pouze jeden nulový endpoint s řídicími přenosy. Všechna data jsou přijímána i odesílána skrz něj.

Audio class používá nulový endpoint pro přenos signálů řídicími přenosy. Druhý používaný endpoint využívá izochronní přenosy.

Communication Device Class (CDC) implementuje dva datové endpointy využívající hromadné přenosy. Jeden vstupní, druhý výstupní. Navíc je k dispozici jeden vstupní endpoint komunikující přerušovanými přenosy.

Z těchto dostupných tříd se k našemu účelu nejvíce hodí třída Communication Device Class, přičemž využijeme její vstupní endpoint s přerušovanými přenosy a jeden výstupní s hromadným přenosem dat. Zbylý jeden endpoint zůstává nevyužit, tudíž není uveden ani v deskriptoru a celá funkcionalita třídy CDC, která bude ve výsledku upravena, s ním nebude počítat. Vzniklé deskriptory pro výsledné USB zařízení jsou vidět ve výpisech 3, 4, 5.

```
Device Descriptor:
----------------------------
0x12  bLength
0x01  bDescriptorType
0x0200  bcdUSB
0x00  bDeviceClass
0x00  bDeviceSubClass
0x00  bDeviceProtocol
0x40  bMaxPacketSize0  (64 Bytes)
0x17E9  idVendor
0x029D  idProduct
0x0200  bcdDevice
0x01  iManufacturer  "DisplayLink"
0x02  iProduct  "Lenovo LT1421 wide"
0x03  iSerialNumber  "6V9CHDG9"
0x01  bNumConfigurations
```

Výpis 3: Device deskriptor vzniklého zařízení

```
Configuration Descriptor:
-----------------------------
0x09 bLength
0x02 bDescriptorType
0x0037  wTotalLength  (55 Bytes)
0x01 bNumInterfaces
0x01 bConfigurationValue
0x00 iConfiguration
0xC0 bmAttributes  (Self-powered Device)
0x32 bMaxPower  (100 mA)
```

Výpis 4: Konfigurační deskriptor vzniklého zařízení

```
Interface Descriptor:
-----------------------------
0x09 bLength
0x04 bDescriptorType
0x00 bInterfaceNumber
0x00 bAlternateSetting
0x02 bNumEndPoints
0xFF bInterfaceClass  (Vendor specific)
0x00 bInterfaceSubClass
0x00 bInterfaceProtocol
0x00 iInterface
```

Výpis 5: Deskriptor rozhraní vzniklého zařízení

4.3.3 Základní konfigurace a vytvoření projektu

Jak již bylo zmíněno v kapitole 3.1.1, STMicroelectronics poskytuje ke svým produktům řadu softwaru pro usnadnění vývoje. Jedním z nich je i STM32Cube MX. Jedná se o grafický konfigurační software, který usnadňuje počáteční konfiguraci a nastavení projektu pro daný mikropočítač. Po výběru procesoru a kitu, pro který chce uživatel vygenerovat projekt, může za pomocí grafického rozhraní konfigurovat výstupy, časování a frekvence, dostupné periferie (v našem případě hlavně USB), atd.

Po dokončení potřebného nastavení je možno vygenerovat projekt pro jedno z dostupných vývojových prostředí (v našem případě Atollic TrueSTUDIO).

STM32CubeMX tedy po zvolení správného vývojového kitu a mikroprocesoru zobrazí předdefinované nastavení, které je možno upravit. Bylo zapotřebí zvolit takové nastavení, aby se výsledné zařízení chovalo v režimu USB Device na svém microUSB portu. Dále bylo povoleno

napájení ze sběrnice a zvolena třída Communication Device Class jako middleware. Za pomocí grafického rozhraní bylo nakonfigurováno VendorID, ProductID a ManufacturerString. Samotný kit podporuje pouze USB standard FullSpeed. Standardu HighSpeed by mohlo být dosaženo za pomocí externího PHY, což je samostatný obvod, který hraje roli můstku mezi digitálními a modulovanými částmi rozhraní.

Dále bylo zapotřebí nastavit LCD displej. V periferiích je nutné aktivovat DMA2D. V dalším kroku je nutno FMC nastavit SDRAM 1 Clock and chip enable na SDCKE1+SDNE1, Internal bank na hodnotu 4 banks, adresu 12 bit, Data 16 bit a povolit 16 bit byte. Nastavení pokračovalo vybráním sběrnice I2C u periferie I2C3, zvolením typu displeje RGB666 (18 bit) u LTDC a u periferie SPI5 navolením modu Full-Duplex Master.

V záložce Clock Configuration bylo ještě závěrem nutno zvolit hodnotu u Main PLL–N na X 336 a u PLLSAI–N na X 432.

Touto konfigurací za pomocí softwaru STM32Cube MX byl vytvořen základní projekt, který obsahuje vše potřebné pro vývoj. Obsahuje CDC jako Middleware, má vložené potřebné knihovny pro práci s displejem a USB rozhraním. Je tedy připraven na import do vývojového studia.

4.3.4 Inicializace integrovaného LCD displeje

Projekt vygenerovaný z STM32Cube MX je spustitelný a již plně funkční, neobsahuje ovšem potřebné knihovny po správnou funkci vestavěného LCD displeje. Díky různým konfiguračním wizardům byly sice nastaveny patřičné hardwarové součásti, avšak v projektu chybí knihovna pro práci s integrovaným displejem. Je proto potřeba ji ručně naimportovat a také projektu přidat další cestu pro include hlavičkových souborů. Potřebné zdrojové soubory se nacházejí v repozitáři softwaru STM32Cube MX. Je tedy vhodné si tuto složku překopírovat do svého projektového adresáře. Konkrétně do podsložky Drivers.

Importovat je potřeba tyto soubory z adresáře: *../Drivers/BSP/STM32F429I–Discovery/*.

- stm32f429i_discovery.c,
- stm32f429i_discovery_lcd.c,
- stm32f429i_discovery_sdram.c.

A ještě soubor ili9341.c z adresáře:
../Drivers/BSP/Components/ili9341/.

Dále je zapotřebí uvést správnou cestu pro hlavičkové soubory:
../../Drivers/BSP/STM32F429I–Discovery/

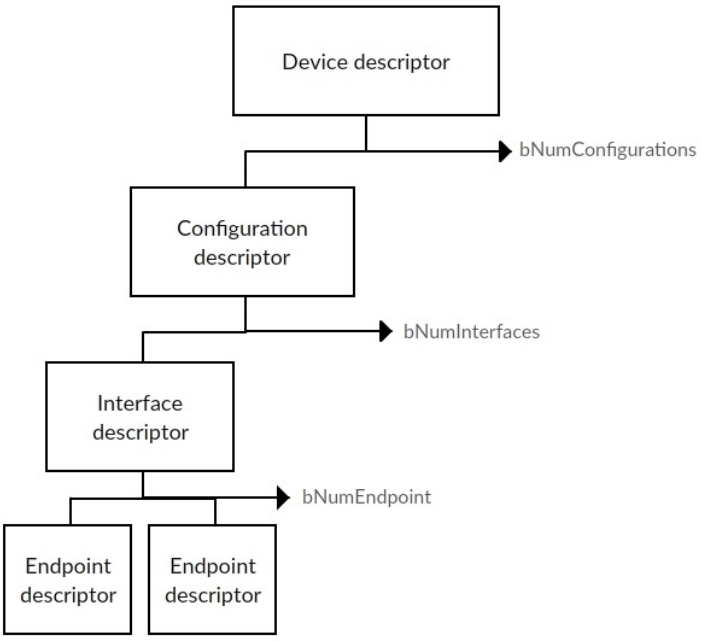
Po úspěšné konfiguraci je ještě třeba zavolat ve funkci **main** metody **BSP_LCD_Init()** a **BSP_LCD_DisplayOn()** pro incializaci a zapnutí integrovaného LCD displeje.

4.3.5 Konfigurace deskriptoru

Třída CDC, kterou lze vygenerovat z STM32Cube MX pracuje s protokoly “Universal Serial Bus Class Definitions for Communications Devices Revision 1.2 November 16, 2007"a specifikací podprotokolu “Universal Serial Bus Communications Class Subclass Specification for PSTN Devices Revision 1.2 February 9, 2007". [8] Tyto vygenerované protokoly bylo třeba upravit, aby výsledná třída mohla spolupracovat s ovladačem DisplayLink. Také bylo zapotřebí upravit deskriptory zařízení, které byly vygenerované pro plnou podporu CDC za všech dostupných rychlostí komunikace.

Díky odposlechnuté komunikaci a získanému deskriptoru za pomocí programů USBlyzer [12] a Thesyncon USB Descriptor Dumper [13] byl zpětně sestaven vhodný deskriptor pro vznikající USB zařízení a upraveny funkce, které provádějí inicializaci třídy CDC, aby komunikace probíhala obdobně jako u připojeného LCD Displeje LT1421.

Vzhledem k tomu, že vývojový kit má omezení pouze na standard FullSpeed, postačuje nakonfigurovat pouze jeden deskriptor, který představuje veškerou konfiguraci zařízení potřebnou pro komunikaci v režimu FullSpeed. Jedná se tedy o jeden konfigurační deskriptor, jeden deskriptor rozhraní a dva deskriptory endpointů (struktura deskriptorů viz obrázek 7). Jeden endpoint ve vstupním režimu IN s přerušovanými přenosy a druhý výstupní endpoint v režimu OUT s hromadnými přenosy.



Obrázek 7: Struktura deskriptorů USB zařízení

4.4 Inicializační sekvence

Pro správnou funkci zařízení bylo nutné upravit inicializaci implementované třídy Communication Device Class, která byla použita jako výchozí bod pro vývoj. Musely být upraveny velikosti paketů u jednotlivých endpointů a endpoint využívající hromadné přenosy ve směru k USB Hostovi (IN) byl úplně zrušen, jelikož jej DisplayLink nijak nevyužívá.

4.5 Enumerace zařízení

Po připojení zařízení do USB portu USB Host rozpozná, že bylo připojeno nějaké zařízení. To je zajištěno díky pull–up rezistoru, který je umístěn na jednom z datových vodičů. Podle toho, zda pull–up rezistor spojuje vodič D+ nebo D– s napětím 3.3 V se určuje rychlost zařízení (ve spojení s D+ se jedná o FullSpeed zařízení a s D– jde o LowSpeed). USB Host provede reset zařízení. Toto zařízení pak dostane výchozí adresu 0. Host pošle požadavek na endpoint 0 s adresou 0, pro zjištění největší možné velikosti paketu. Tento požadavek se provádí za použití příkazu GetDescriptor. Po obdržení deskriptoru zařízení je typicky proveden další reset a příkazem SetAddress je přiřazena unikátní adresa zařízení, které mělo doposud výchozí adresu 0. Host dále získává více informací o zařízení posíláním požadavků:

- Get Device Descriptor,
- Get Configuration Descriptor,
- Get String Descriptor.

Zařízení je v tomto stavu adresováno, avšak není nakonfigurováno, takže je schopno odpovídat pouze na standardní požadavky. Jakmile USB Host získá dostatek informací o zařízení, načte pro něj vhodný ovladač. Ten pak zvolí požadavkem SetConfiguration patřičnou konfiguraci zařízení, o které rozhodne na základě získaných informací z obdržených deskriptorů.

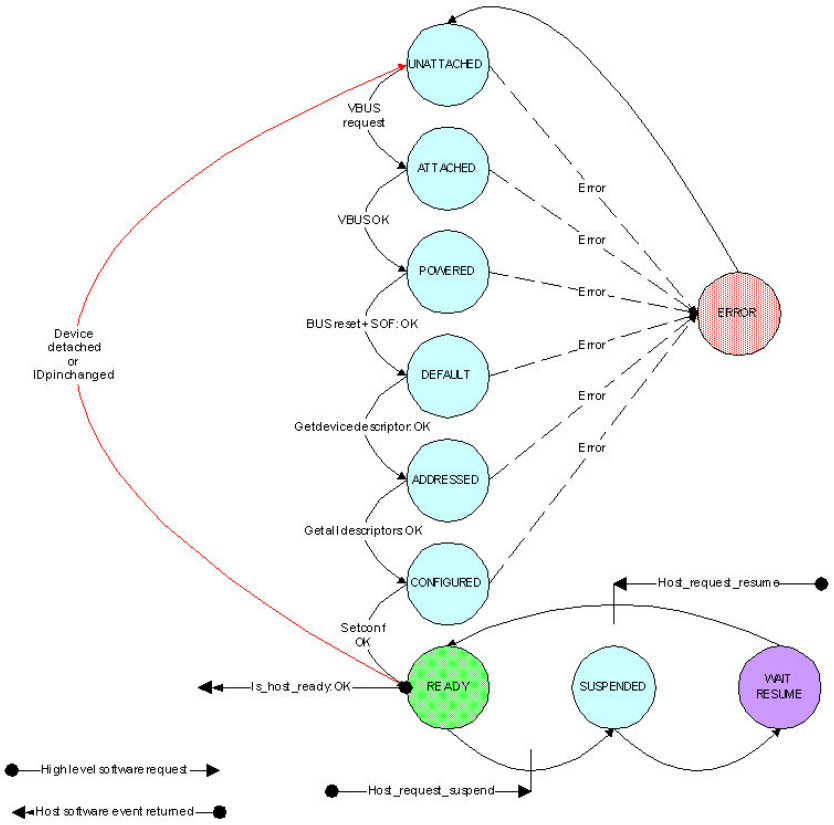
USB zařízení se takto dostane do nakonfigurovaného stavu a může pracovat způsobem, jakým bylo navrženo. Nyní je zařízení schopno reagovat mimo standardních i na specifické požadavky a komunikovat pomocí nakonfigurovaných endpointů různými typy přenosů.

Do doby, než je zařízení v nakonfigurovaném stavu, je s ním možné komunikovat pouze řídicími přenosy po výchozím endpointu 0, kterým lze data přenášet obousměrně.[16]

Proces enumerace je v jednotlivých krocích znázorněn na stavovém diagramu 8. Diagram znázorňuje i stav Error, do kterého se proces dostane v případě, že nastane nějaká chyba. Z tohoto stavu se USB zařízení dostane to nepřipojeného stavu a zkouší provést enumeraci odznova.

4.6 Úprava enumerace

Aby zařízení bylo správně rozpoznáno jako displej DisplayLink, bylo nutné upravit jeho deskriptor. Samotná úprava deskriptoru je popsána v kapitole 4.3.5. USB Host po změně deskriptoru je schopen zařízení rozpoznat a přiřadit mu adresu.



Obrázek 8: Stavový diagram procesu enumerace [21]

Při průběžném monitorování provozu mezi USB Hostem a zařízením nyní byl zjištěn mimo standardních požadavků, jenž jsou podrobněji popsány v kapitole 4.5, také požadavek, který se v nástroji USBlyzer tvářil jako „Vendor Device“ s jedním parametrem (číselnou hodnotou) 0x06. Tento požadavek knihovna USB na zařízení není schopna rozpoznat a tak na něj odpovídá chybovou hláškou **Stall(PID)**. Do rozhodovacího algoritmu, který je tvořen přepínačem, kde jednotlivé případy tvoří standardní požadavky pro komunikaci přes endpoint 0, byl tedy přidán nový případ s tělem, který odpovídá stejnou sekvencí bajtů, jakou na tento požadavek odpovídal displej LT1421. Tyto bajty jsou: **0x01 0x40 0x00 0xF1**. Jedná se pravděpodobně o nějaký druh potvrzovací sekvence.

Po odeslání této sekvence pokročila enumerace do stavu nastavení konfigurace na zařízení příkazem SetConfig, přičemž tento příkaz nese parametr 1, což znamená, že má být zvolena konfigurace s indexem 1 (Tato hodnota je mimo jiné uvedena v konfiguračním deskriptoru. Pokud je konfigurací více, zvolí se konfigurace podle indexu předaného v parametru požadavku. Pokud je konfigurační deskriptor jeden, to znamená, že konfigurace je jen jedna, pak se zvolí tato samotná konfigurace.[16]). Po nastavení konfigurace následuje ještě několik požadavků „Vendor Device“ se stejným parametrem jako v předchozích případech. Zařízení na tento požadavek však po určitém kroku začne odpovídat jinak. Přesněji sekvencí: **0x01 0x50 0x00 0xF1**. Hodnoty

parametrů v požadavku „Vendor Device“ se však nijak nemění. To naznačuje, že je v zařízení DisplayLink využito řešení, které se podobá stavovému automatu. Jednotlivé požadavky, které byly zjištěny během analýzy, jsou vypsány v tabulce 2.

Tabulka 2: Přehled zjištěných požadavků zařízení DisplayLink

Hodnota	Význam
0x02	Čtení EDID
0x05	Neznámý (vyžaduje odpověď sekvencí 0xA4 0x0D 0x00 0x00)
0x12	Nastavení a přijetí šifrovacího klíče
0x14	Potvrzení přijetí hromadného přenosu (vyprázdnění bufferu)
0x15	Pravděpodobně požadavek na deinicializaci

Aby bylo docíleno stejného chování, bylo po rozsáhlejší analýze zjištěno, že změna nastane vždy po určitém počtu obdržených požadavků „Vendor Device“. Toto chování bylo vytvořeno za pomocí čítače, umístěného do dříve vloženého případu. Při každém vykonání je tato proměnná inkrementována a na základě její aktuální hodnoty je za pomocí podmínek zvolena správná sekvence bajtů, která má být na obdržený požadavek USB Hostovi odpovězena.

Po nastavení požadované konfigurace lze na výpisu USB provozu zpozorovat přenos dat na endpoint 0 o velikosti 16 bytů ve směru do zařízení. Těchto 16 bajtů je pravděpodobně využito jako klíč k šifrování dat po „rourách“ (pipe) využívající hromadné přenosy. Blíže je tento proces popsán v kapitole 4.9. Tento přenos je následován dalšími požadavky „Vendor Device“ s parametrem 0x02, na které monitor Lenovo LT1421 odpovídá sekvencí dlouhou 130 bajtů, která, jak bylo po analýze zjištěno nese EDID, neboli informace o monitoru. Popis dat, přenesených v těchto 130 bajtech a jejich cílové určení a účel je vysvětlen v následující kapitole.

```
0x   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -------------------------------------------------
00 | 00 FF FF FF FF FF FF 00 30 AE 21 14 4E 61 BC 00
01 | 17 19 01 03 81 1F 11 78 2A 9F E5 96 58 53 8A 26
02 | 24 50 54 00 00 00 01 01 01 01 01 01 01 01 01 01
03 | 01 01 01 01 01 01 20 1C 56 86 50 00 20 30 0E 38
04 | 13 00 35 AE 10 00 00 1E 00 00 00 FF 00 36 56 39
05 | 43 48 44 47 39 0A 20 20 20 20 00 00 00 FD 00 32
06 | 4B 1E 53 09 00 0A 20 20 20 20 20 20 00 00 00 FC
07 | 00 4C 65 6E 6F 76 6F 20 4C 54 31 34 32 31 00 40
```

Obrázek 9: Originální EDID z Lenovo ThinkVision LT1421

4.7 EDID

Extended Display Identification Data, neboli zkráceně EDID jsou data, která obsahují popisné informace, včetně popisu vlastností monitoru. Nesou například informace o výrobci a modelu, maximálním rozlišení, obnovovací frekvenci, gama atd. Tento datový blok slouží k jednoznačné identifikaci zobrazovacího zařízení a k získání vstupních parametrů pro jeho optimální nastavení,

aby se předešlo nevhodnému nastavení, které by mohlo mít za následek snížení obrazové kvality či pokles stability operačního systému. [17]

Za pomocí programu Deltacast E-EDID editor[18] byla identifikační data upravena pro displej o rozlišení 320×240 pixelů. Přesněji na rozlišení 640×480 pixelů, jelikož se jedná o nejmenší podporované rozlišení technologií DisplayLink. Původní a upravený stav sekvence EDID je patrný z obrázků 9 a 10.

Do zařízení tedy budou posílána data o rozlišení 640×480 pixelů, po přijetí budou redukována na rozlišení 320×240 pixelů za pomocí zprůměrování každé ze tří barevných složek ze čtyř okolních hodnot do jednoho pixelu.

```
0x   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -------------------------------------------------
00 | 00 FF FF FF FF FF FF 00 30 AE 21 14 4E 61 BC 00
01 | 17 19 01 03 81 05 04 78 2A 9F E5 96 58 53 8A 26
02 | 24 50 54 00 00 00 01 01 01 01 01 01 01 01 01 01
03 | 01 01 01 01 01 01 20 1C 40 86 10 F0 20 00 0E 38
04 | 13 00 32 25 00 00 00 1E 00 00 00 FF 00 36 56 39
05 | 43 48 44 47 39 0A 20 20 20 20 00 00 00 FD 00 32
06 | 4B 1E 53 09 00 0A 20 20 20 20 20 20 00 00 00 FC
07 | 00 4C 65 6E 6F 76 6F 20 4C 54 31 34 32 31 00 99
```

Obrázek 10: Upravený EDID pro STM32F429 Discovery

Během analýzy bylo zjištěno, že LCD displej Lenovo LT1421 se při odesílání sekvence EDID chová jinak na Linuxu a jinak na systému Microsoft Windows.

EDID je USB Hostovi odeslán jakmile se dotáže požadavkem „Vendor Device“ s parametrem 0x12.

Na systému Linux je EDID, mající délku 128 bajtů posílán bajt po bajtu směrem k USB hostovi na nulový endpoint. Každá část je doprovázena prefixem 0x00. 128 krát se tedy odešlou dva bajty jako odpověď na požadavek s parametrem 0x12, z nichž první má vždy hodnotu 0x00 a druhý nese patřičnou hodnotu části EDID sekvence. Tento proces je implementován cyklem for, který prochází statickou kolekci, kde je uložen EDID. Cyklus při každém průchodu inkrementuje index, a patřičnou hodnotu přiřadí do nově vytvořeného pole o dvou prvcích. První prvek má hodnotu 0 a druhý prvek nese hodnotu získanou na základě indexu z pole s EDID daty. Toto dvouprvkové pole je v každé iteraci odesláno na endpoint 0 k USB hostovi.

Na systému Microsoft Windows je proces přenosu EDID sekvence mírně modifikovaný. Každá odpověď na požadavek začíná také prefixem 0x00 a za ním následuje patřičná část dat. Nejprve je odesláno prvních 63 bajtů (+ jeden bajt, který zabírá prefix), USB Host se poté dotáže opětovně stejným požadavkem na zbytek sekvence. Logika postupného odesílání je opět implementována přepínačem se třemi případy. Každý požadavek „Vendor Device“ s parametrem 0x12 obsahuje totiž ještě hodnotu, která se vždy v těchto případech liší. Za pomocí přepínače na základě detekce hodnot v požadavku zařízení rozhodne, která část má být odeslána. Ve druhém případě

přepínače zařízení pošle dalších 63 bajtů a jakmile je dotázán na třetí případ, odešle poslední 2 bajty.

Po dokončení tohoto procesu následuje sekvence 4260 bytů dat, která obsahuje Huffmanův binární strom. Následující data jsou již přenášena hromadnými přenosy na druhý endpoint (OUT). Je proto potřeba na tomto endpointu naslouchat a obdržená data odtud včas vyzvednout a zpracovat,aby komunikace probíhala co možná nejrychleji.

4.8 Šifrování komunikace

Jak píší ve svém článku[10] Echtler a Hodges, veškerá komunikace hromadnými přenosy je šifrovaná. Toto šifrování je prováděno na základě klíče (16 bajtů), který je předán v prvním přenosu ve směru OUT. Z tohoto klíče se vypočítá kontrolní redundantní součet CRC12 (za využití polynomu pro generaci: 0x180F = 0001 1000 0000 1111 = $x^{12} + x^{11} + x^3 + x^2 + x + 1$ (standardní CRC12)). Získaná 12 bitová hodnota je pak kombinovaná s příchozími daty bit po bitu za pomocí logické operace XOR.

Jsou však výjimky, kdy k šifrování nedochází. Pokud USB Host odešle tzv. null klíč 11, data přenesená hromadnými přenosy ve směru OUT by neměla být zašifrovaná.

Šifrování na zařízeních DisplayLink pravděpodobně zajišťuje obvod posuvného registru s lineární zpětnou vazbou (LFSR), který se stará o to, aby byl jeho výstup lineárně závislý na jeho předchozích výstupech a stavu. Toto technické řešení má výhodu v tom, že se jedná o hardwarovou součástku. Díky tomu nesnižuje výkon a šetří paměť celé jednotky, na které je umístěno. Významným kladem je také fakt, že zašifrování i rozšifrování se provádí stejným postupem, stačí tedy navrhnout jednu část a druhá je automaticky totožná. Účinnost šifrování, které je prováděno logickými operacemi spočívá v náhodně generovaném klíči. Problémem však může být distribuce klíče na druhou stranu, pro účely dešifrování dat.

- 47 3d 16 97 c6 fe 60 15 5e 88 1c a7 dc b7 6f f2
- 57 cd dc a7 1c 88 5e 15 60 fe c6 97 16 3d 47 f2

Obrázek 11: Null klíče, které zamezí šifrování

4.9 Huffmanova komprese

Pro zmenšení objemu přenášených dat technologie DisplayLink využívá Huffmanova kódování, které se řadí mezi bezztrátové komprimační metody. Na základě provedené analýzy bylo zjištěno, že komprese je využita pouze u ovladače pro systém Microsoft Windows a Mac OS. Systém Linux ve svém ovladači **udlfb** komprimační algoritmus nevyužívá. V této podkapitole je popsán princip komprese a postup jak data dekomprimovat. Možnost rozšíření produktu této diplomové práce o kompresní metodu, tak aby bylo možné LCD displej plně využívat i na Systémech společnosti Microsoft je rozepsána v kapitole 6.

Tato kompresní metoda konvertuje znaky vstupního souboru do bitových řetězců různé délky. Znaky, které se ve vstupním souboru vyskytují nejčastěji, jsou konvertovány do bitových řetězců s nejkratší délkou (nejfrekventovanější znak tak může být konvertován do jediného bitu) a znaky, které se vyskytují velmi zřídka, jsou konvertovány do delších řetězců (mohou být i delší než 8 bitů). Za pomocí sestaveného Huffmanova stromu jsou data zakódována a zpravidla společně se stromem odeslána k adresátovi. Dekomprese je prováděna za pomocí rekonstruovaného binárního Huffmanova stromu, který je sestaven z přijatých bitů. Přijatá data jsou na základně bitové cesty k listu zaměněna za hodnotu, kterou obsahuje daný list.

4.9.1 Sestavení Huffmanova stromu

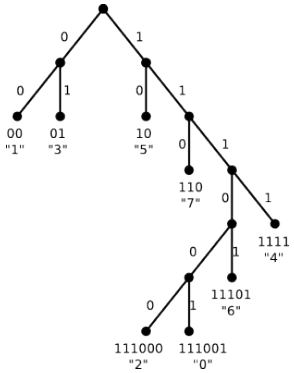
Kódování je prováděno vyhledáváním znaku ve stromě a zapamatováním si uražené cesty od kořene k nalezenému listu. Požadované původní znaky jsou uloženy v listech stromů. Krok k levému uzlu znamená zaznamenání 0 a krok k pravému uzlu 1. Algoritmus vždy vyhledává od kořene stromu. Pokud tedy během vyhledávání některého ze znaků byly navštíveny čtyři uzly (včetně kořene a listu stromu) a cesta byla: doprava, doprava, doleva – pak je zaznamenaná cesta: 110. Tato cesta je použita k nahrazení znaku, který je uložen v listu stromu, v původních nekomprimovaných datech a tím dojde ke kompresi, protože znak s nejvyšší četností výskytu může být nahrazen například i jedním bitem. Příklad Huffmanova stromu je znázorněn na obrázku 12

Takto sestavený strom je odeslán zpravidla spolu s komprimovanými daty na místo doručení a tam je následně využit k dekompresi přijatých dat. Huffmanův strom může být poměrně rozsáhlý (záleží například na spektru užitých znaků), je tedy vhodné tuto kompresi využít v případech, kdy odesíláme větší množství dat, či jej budeme opakovaně využívat k dekompresi přijatých dat po nějakou dobu. Může se totiž stát že by Huffmanův strom měl větší velikost než samotná přijatá data. Tím by celá komprese ztrácela význam a byla by extrémně neefektivní.

U technologie DisplayLink se o neefektivní využití nejedná, jelikož Huffmanův strom je odeslán na počátku celé komunikace a po celou dobu je využíván jeden a ten samý strom, který byl přijat při inicializaci. Navíc jeho velikost (4620 bajtů) nepřesahuje velikost dat, která jsou odeslána do zařízení DisplayLink v rámci inicializace (tato velikost dat se však mění s typem zařízení a jeho rozlišením).

4.9.2 Dekomprimace

Přijatý strom je na cílové straně využit k dekomprimaci za použití reverzního způsobu jakým byla data zakódována (nahrazení komprimovaných sekvencí znaky z listů stromu). Ke správnému sestavení stromu je nutno obdržet data binárního stromu a vědět kolik bitů je určeno pro list a kolik pro cestu.



Obrázek 12: Příklad Huffmanova binárního stromu

4.10 Příkazy DisplayLink

Komunikace přes hromadné přenosy na endpointu 1 je doprovázena příkazy, kterými USB host displeji podporujícímu protokol DisplayLink sděluje, jak má s daty naložit.

Všechny příkazy, u kterých bylo zjištěno, že se v komunikaci vyskytují, začínají jedním bajtem s hodnotou **0xAF**. Za ním následuje další bajt s hodnotou, určující jak budou data zpracována. Tato hodnota jednoznačně identifikuje typ dat. Konkrétní hodnoty jsou uvedeny v tabulce 3. Tato tabulka znázorňuje i formát, v jakém jsou data předávána.

Po těchto dvou bajtech obvykle následuje sekvence dat.

U příkazů, které pracují s registry je místo sekvence umístěn jeden bajt jako adresa a jeden jako hodnota, která má být na danou adresu do registru uložena.

Příkazy ukládající, nahrazující či jinak pracující s pamětí jsou následovány třemi bajty nesoucí offset, dále jedním bajtem obsahujícím celkový počet dat a pak sekvencí dat, která se skládá ze dvojic (počet, hodnota).

Požadavky, nesoucí komprimovaná obrazová data mají po hodnotě 0x70 nebo 0x78 (záleží na typu grafických dat, zda jsou 8-bitová či 16-bitová) vždy tři bajty obsahující offset, dále jeden bajt obsahující počet pixelů a pak komprimovaná obrazová data.

4.11 Příjem dat

Využitím třídy CDC bylo docíleno požadované komunikace hromadnými přenosy po endpointu 1. Třída již obsahuje připravenou kostru implementace pro příjem a odesílání dat. Bylo třeba zvolit vhodný způsob zpracování dat. Po analýze problematiky bylo zvoleno řešení uložit všechna přijatá data do kolekce typu FIFO, která bude implementována formou kruhového bufferu a ta bude poté poskytována dalším metodám ke zpracování. Tato kolekce bude samozřejmě průběžně vyprazdňována.

Paměťové nároky však s implementací této kolekce rostou i s přihlédnutím že při následném zpracování přijatých dat bude zapotřebí velké části paměti pro uložení naparsovaných dat před zobrazením. Prozkoumáním možností bylo zavrženo řešení za použití **malloc** a využívání paměti

Tabulka 3: Přehled zjištěných příkazů zařízení DisplayLink

Hodnota	Význam	Formát
0x20	Zápis do registru	0xAF 0x20 1_adresa 1_hodnota
0x40	Neznámý	
0x60	Zápis do paměti	0xAF 0x60 3_offset 1_delka N_data
0x61	Vyplnění paměti	0xAF 0x61 3_offset 1_celk_delka N_data [1_pocet 1_hodnota]
0x62	Kopírování paměti	0xAF 0x62 3_cil_offset 1_pocet 3_zdroj_offset
0x63	Nahrazení v paměti	0xAF 0x63 3_cil_offset 1_pocet 3_zdroj_offset
0x67	Neznámý	
0x68	Zápis do paměti	0xAF 0x68 3_offset 1_delka N_data
0x69	Vyplnění paměti	0xAF 0x69 3_offset 1_celk_delka N_data [1_pocet 1_hodnota]
0x70	Zápis kompr. dat 8-bit	0xAF 0x70 3_offset 1_pix_pocet N_data
0x78	Zápis kompr. dat 16-bit	0xAF 0x78 3_offset 1_pix_pocet N_data
0x6A	Kopírování paměti	0xAF 0x6A 3_cil_offset 1_pocet 3_zdroj_offset
0x6B	Nahrazení paměti	0xAF 0x6B 3_cil_offset 1_pocet 3_zdroj_offset
0x6E	Neznámý	
0xA0	Čistění příkazového bufferu	0xAF 0xA0
0xE0	Huffmanův strom	0xAF 0xE0

RAM. Vývojový kit má k dispozici SDRAM, přičemž je využívána jen z části LCD displejem, který zde má alokované místo pro uložení jednotlivých obrazových vrstev. Byla tedy alokována paměť za tímto místem.

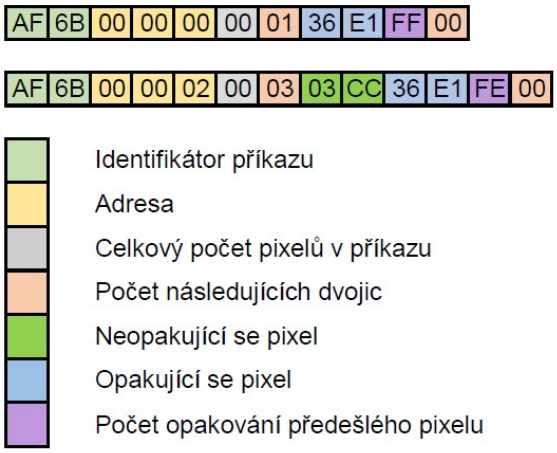
Příjem dat se uskutečňuje ve zdrojovém souboru **usbd_cdc_if.c**. Jsou zde připraveny funkce pro inicializaci celého rozhraní (vytvoření bufferů, otevření endpointů, navázání spojení, atd.), příjem dat, odeslání dat a deinicializaci rozhraní (uzavření komunikace, uzavření endpointů, atd.). Metoda pro odesílání dat byla v tomto řešení nevyužita, jelikož rozhraní původní třídy CDC bylo upraveno tak, aby sloužilo jen pro příjem, tedy obsluhovalo pouze jeden endpoint. Naopak velice důležitou roli zde hraje metoda **USBD_CDC_ReceiveHS()**, která obstarává příjem dat. Vstupními parametry jsou ukazatele na buffer a na délku dat uložených v bufferu. Pro uschopnění procesu příjmu dat bylo zapotřebí ještě umístit těsně před návratovou hodnotu do metody pro inicializaci volání funkce *USBD_CDC_ReceivePacket()* pro přijetí prvního paketu z bufferu od USB Hosta.

Data jsou přijímána po 64 bajtech (velikost paketu u standardu USB FullSpeed). Algoritmus tedy cyklem while zpracuje pakety a přidá je do zmiňované kolekce, která je kruhovým bufferem. Po zpracování každého paketu je vždy volána funkce *USBD_CDC_ReceivePacket()*, která zajistí přijetí dalšího paketu do pracovního kruhového bufferu. Díky tomu může být zpracován další přijatý paket. Po dokončení procesu přijímání dat následuje proces, který přijatá data naparsuje, resampluje a zobrazí. Volání funkcí které obstarávají tento proces je umístěn v nekonečném cyklu ve zdrojovém souboru **main.c** a kruhový buffer je tímto způsobem postupně vybírán.

4.12 Parsování přijatých dat

Originální ovladač DisplayLink využívá poměrně rozsáhlou skupinu příkazů pro přenos dat ke zobrazení z USB Hosta ke klientovi. Jedná se například o příkazy, které nesou komprimovaná data, nekomprimovaná data, příkazy pro nahrazení určité části obrazovky, příkazy pro kopírování atd. Všechny tyto příkazy jsou popsány v tabulce 3.

Ovladač pro Linux **udlfb** využívá pro přenos obrazových dat jediný příkaz **0xAF 0x6B**. Je to převážně z důvodu, že tento ovladač je odlehčenou verzí originálního ovladače a nemá natolik propracované mechanizmy, aby dokázal vytvořit například komprimovaná data.



Obrázek 13: Struktura sekvence AF 6B

Celá struktura výše zmíněného příkazu je započata prefixem **0xAF 0x6B**. Za ním následují tři bajty, které určují, kam data přijdou umístit (ukazují na místo v paměti framebufferu). Poté jsou v sekvenci přeneseny dva bajty. První z nich označuje celkový počet pixelů, které jsou tímto příkazem přenášeny (maximální počet přenesených pixelů v jednom příkazu je 256). Druhý pak určuje kolik dvojic bajtů za tímto bajtem následuje bez přerušení. Přerušením máme na mysli dodatečný bajt, jehož hodnota udává počet opakování poslední přečtené dvojice. Zmíněné dvojice bajtů reprezentují jeden obrazový bod (pixel) ve formátu 5–6–5. Červenou barvu zastupuje pět, zelenou šest a modrou pět bitů. Jednotlivá obrazová data jsou přenášena po řádcích a jak již bylo zmíněno, jeden příkaz může obsahovat až 256 obrazových bodů. Pixel ve kterém začíná sekvence bodů udává adresa přenesená hned za prefixem příkazu. Adresy dalších pixelů, které jsou přenášeny ve stejném příkazu jsou inkrementovány. Pixely jsou tedy umisťovány jeden vedle druhého zleva doprava. V případě, že má více za sebou přenášených bodů stejné barevné hodnoty (například u jednobarevného pozadí), dochází při přenosu k jakési jednoduché kompresi. Bajt, který nese informaci o počtech dvojic bez přerušení by pak měl například hodnotu jedna (pokud by všech 256 pixelů mělo stejnou barvu) a za ním by byla pouze jedna dvojice. Ihned za touto dvojicí je v případě opakování umístěn další bajt, který udává počet, kolikrát se má předešlý obrazový bod v řádku opakovat. Ukázky bajtových sekvencí jsou vyobrazeny na obrázku 13.

Každá sekvence končí bajtem s hodnotou 0x00. Jedná se o bajt určující kolik dvojic bajtů za ním následuje. Hodnota je 0, tedy žádné další pixely už za tímto bajtem nejsou, algoritmus očekává další příkaz.

Přijatá data jsou pak vykreslována tak, že k jednorozměrnému poli, které obsahuje přijatá data je přistupováno jako ke dvourozměrnému pomocí vzorce $y \times A + x$, kde x a y jsou horizontální a vertikální souřadnice a A je šířka obrazu.

Algortimus, který provádí parsování přijatých dat je patrný z výpisu 7.

4.13 Resampling obrazových dat

Jak již bylo zmíněno v kapitole 4.7, kde je popsána tvorba sekvence EDID a její vliv na zobrazovací zařízení, nejnižší rozlišení, které je podporováno ovladačem **udlfb** je 640×480, avšak LCD displej na vývojovém kitu STM32F429 DISCOVERY podporuje vzhledem ke své velikosti maximální rozlišení 320×240.

Tento problém je řešen redukcí většího rozlišení na menší (resamplováním). Přijatá a zpracovaná data jsou uložena do jednorozměrného pole o velikosti podporovaného rozlišení. Ukládána jsou již v barevném formátu 8–8–8–8, který podporuje i alfa kanál, ačkoliv z ovladače mají získaná data barevný formát 5–6–5.

Pro získání přibližných souřadnic pro zmenšený obraz je vypočítán poměr vstupní a výstupní výšky a šířky obrazu. Kolekce je procházena dvěma do sebe zanořenými for cykly a pro každý bod nového obrazu je vypočítána souřadnice pixelu v jednorozměrném poli, který je nejblíže v původním obraze. Tato hodnota je okamžitě vykreslena za pomocí funkce ***BSP_LCD_DrawPixel()*** na integrovaný LCD Displej. Kód tohoto postupu je na výpise 6

Tento krok je proveden vždy po přijetí a zpracování příkazu **AF 6B**. Jakmile je daná hodnota zobrazena na displeji, je držena v paměti do té doby, než ji nahradí hodnota jiná.

Resampling lze vyřešit také postupem nazývaným interpolace, při kterém dochází ke zprůměrování okolních hodnot bodu. Například se sečtou hodnoty troj-okolí konkrétního bodu a vydělí se celkovým počtem, tedy devíti.

Funkce ***BSP_LCD_DrawPixel()*** je součástí knihovny od STMicroelectronics, která je dodávaná s firmwarem k vývojovému kitu. Jedná se soubor funkcí a nástrojů, které umožňují pracovat s integrovaným LCD displejem. Zmíněná funkce jako parametry přijímá souřadnici **x**, **y** a hodnotu pixelu (barva ve formátu 8–8–8–8). Dále jsou k dispozici metody pro obsluhu LCD displeje, vykreslování různých tvarů (obdélník, elipsa, trojúhelník, polygon), linek, textových znaků, umisťování těchto elementů do vrstev atp.

```
void Resample()
{
    double scaleWidth = (double)newWidth / (double)width;
    double scaleHeight = (double)newHeight / (double)height;

    int cy;
    int cx;

    for (cy = 0; cy < newHeight; cy++)
    {
        for (cx = 0; cx < newWidth; cx++)
        {
            int nearestMatch = (((int)(cy / scaleHeight) * (width)) + ((int)(cx / scaleWidth)));
            BSP_LCD_DrawPixel(cy, cx, svga[nearestMatch]);
        }
    }
}
```

Výpis 6: Redukce rozlišení

```
void Parse (void)
{
   uint32_t *svga = (uint32_t*) SDRAM_800_600_START;
   uint8_t cmdThis = 0, cmdLast = 0;
   uint8_t temp[3];
   while(!RingBuffer_empty(CDCringBuffer))
   {
      RingBuffer_read(CDCringBuffer, &cmdThis, 1);
      if (cmdLast == 175 && cmdThis == 107)
      {
         uint32_t address = 0;
         uint32_t currentPixelValue = 0;
         uint32_t repeatCount = 0;
         uint32_t i;
         RingBuffer_read(CDCringBuffer, temp, 3);
         address = ((((uint32_t)temp[0]) << 16) & 0x00FF0000) \
                 | ((((uint32_t)temp[1]) << 8) & 0x0000FF00)\
                 | (((uint32_t)temp[2]) & 0x000000FF); //read address (start at 0 end at 256 in case of 256
                     pixels)
         address >>= 1; //divide by 2 - former addressing was byte wise, we are pixel wise.

         uint32_t totalPixelCount = 0;// read count of all pixels in this command (256 == 0)
         RingBuffer_read(CDCringBuffer, &totalPixelCount, 1);
         uint32_t rawPixelCount = 0; // read count of pixels to read (byte tuples)
         RingBuffer_read(CDCringBuffer, &rawPixelCount, 1);
         rawPixelCount += 1;
         while (rawPixelCount > 1) // process current number of byte tuples
         {
            RingBuffer_read(CDCringBuffer, &currentPixelValue, 2);
            currentPixelValue = ((currentPixelValue & 0xFF) << 8) | ((currentPixelValue & 0xFF00) >>8);
            uint32_t color = ShortToLong(currentPixelValue);
            svga[address++] = color;
            rawPixelCount -= 2;
            if (rawPixelCount < 1)
            {
               repeatCount = 0;
               RingBuffer_read(CDCringBuffer, &repeatCount, 1);
               for(i=0; i < repeatCount; i ++)
                  {svga[address++] = color;}
               //read count of next bytes for reading
               rawPixelCount = 0;
               RingBuffer_read(CDCringBuffer, &rawPixelCount, 1);
            }
         }
         Resample(svga);
      }
      cmdLast = cmdThis;
   }}
```

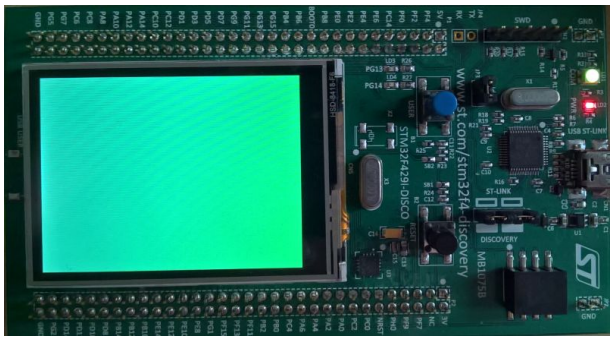
Výpis 7: Parsování příkazu AF 6B

5 Testování

V průběhu vývoje programového vybavení pro modul STM32F429I-DISCOVERY byly již vytvořené části průběžně testovány tak, aby byla zaručena funkčnost aktuálně vyvíjených modulů a tím se eliminovaly skryté chyby které by mohly vzniknout a zůstat skryté hluboko ve funkcionalitě.

V prvním kroku byla tvořena komunikace po USB, kdy šlo především o to aby probíhala správně komunikace se zařízením a veškerá přijatá obrazová data byla prozatím zahazována a nevyužita. Díky tomu, že byl k dispozici originální LCD displej Lenovo LT1421, byla jeho komunikace s USB hostem odposlouchávána a postupně dolaďována k požadované formě. Odchytávat šla také komunikace mezi vyvíjeným modulem a USB hostem. Porovnáváním těchto dvou záznamů bylo možné odhalit vzniklé potíže v komunikaci. Například zobrazením jednobarevného pozadí či různých vzorů nebo třeba změnou rozlišení bylo možné nasimulovat podmínky, které by mohly být pro komunikaci i pro celý proces zobrazení kritickými body.

Častým problémem byla nesprávně přenesená nebo vůbec nepřenesená sekvence EDID. Ovladač **udlfb** pak tento LCD displej sice rozezná, ale nezobrazí na něm žádná data. Vykreslí na všechny pixely zelenou barvu a dostane se do stavu FB_BLANK. Tuto situaci je možno vidět na obrázku 14.

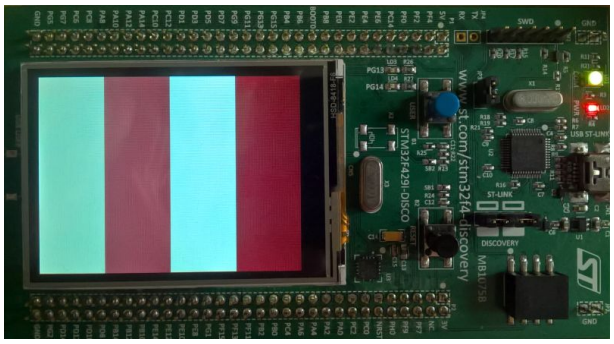


Obrázek 14: Nepřenesená sekvence EDID

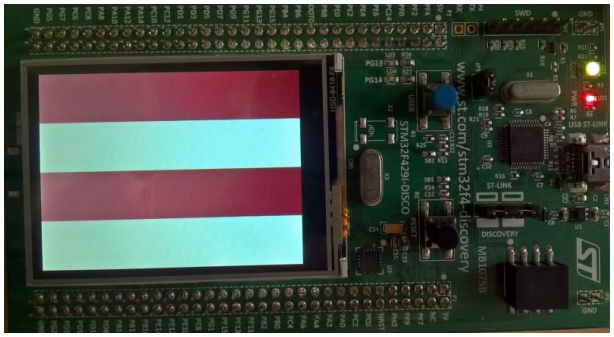
Dalším krokem byl vývoj funkcionality, která obstarává příjem dat do vhodné struktury a po obdržení všech paketů data předává algoritmu pro resampling, neboli přetransformování obrazových dat na menší rozlišení a následné zobrazení těchto dat. Podstatnou roli v této části také hrálo zprovoznění knihovny a nastavení celého modulu pro podporu integrovaného LCD displeje. Zde za využití vykreslování různých jednobarevných pozadí a vzorů s horizontálními 15 a vertikálními pruhy 16 bylo docíleno optimálního softwarového řešení pro redukci rozlišení a také správné orientace displeje. Ve výchozím nastavení je totiž LCD displej orientován nikoli horizontálně, ale vertikálně. Je tedy zapotřebí vždy myslet na prohození souřadnic **x** a **y**.

Vzniklý LCD displej byl také testován na různých operačních systémech Linux. Jelikož ovladače na systémy Windows jsou díky zaměření vývoje především na tyto systémy mnohem so-

fistikovanější, prozatím tento displej není schopen fungovat pod jiným operačním systémem než pod Linuxem. Možnosti tohoto rozšíření jsou popsány v kapitole 6.



Obrázek 15: Test LCD displeje: horizontální pruhy



Obrázek 16: Test LCD displeje: horizontální pruhy

Obrázek 17: Test LCD displeje: jedna barva

6 Možnosti rozšíření a vylepšení

Vzniklý LCD displej je funkční a použitelný v praxi, mnohé je však třeba ještě doladit či rozšířit. Otevírá se zde mnoho směrů, jak tento produkt zdokonalit tak, aby byl opravdu spolehlivý a multiplatformní. Prozatím je funkční na systémech Linux za použití ovladače **udlfb**, jehož implementace je zatím oproti originálnímu ovladači pro Windows značně zredukovaná. Neobsahuje například žádný kompresní algoritmus, díky kterému je možno výrazně zvýšit výkonnost celého řešení, neboť komprimovaná data je možnost rychleji doručit na cílovou stranu a tím tak dosáhnout lepší odezvy celého displeje. Redukce a omezení ovladače pro systémy Linux je zapříčiněna tím, že vývoj je primárně zaměřen pro platformy, na nichž jsou instalovány systémy Microsoft Windows a Mac OS X. Cílovými zákazníky jsou především firmy a společnosti, které DisplayLink technologii využívají v drtivé většině k eliminaci problémů při přenosech pracovních stanic a k podpoře mobility svých zaměstnanců. Ovladač **udlfb** proto také vznikl později a to díky práci Floriana Echtlera, který reverzním engineeringem vytvořil knihovnu **libdlo**, která byla impulsem pro vznik ovladače **udlfb**, který již poté oficiálně zaštítil DispalyLink.

Jeden směr možného vylepšení tedy spočívá v rozšíření stávajícího řešení i pro platformu Windows tak, aby displej využil všechny výhody a možnosti, které ovladače na této platformě poskytují. Toto řešení vyžaduje implementaci dekódování Huffmanova stromu, který je přijat během inicializačních sekvencí, implementaci jeho zpracování a uložení do vhodné datové struktury. Následně by toto rozšíření muselo implementovat algoritmus, který by data přijatá hromadnými přenosy nejdříve dekomprimoval a až poté je předal ke zobrazení na LCD displeji. Popisované rozšíření by zajisté muselo implementovat algoritmus pro dešifrování obdržených dat i přes to, že nejnovější ovladače již šifrování pravděpodobně nevyužívají. Pro zachování zpětné kompatibility by tato podmínka však byla nutná.

Další směr zdokonalení může například vycházet z faktu, že nemáme k dispozici integrovaný LCD displej na dostupném hardwaru nebo chceme použít hardware, na kterém je programové vybavení nahráno, pouze jako adaptér a ne jako zobrazovací jednotka. Šla by například implementovat součást, která by přijatá data posílala na sběrnici SPI, ke které by mohl být připojen LCD displej. Narazili bychom zde však na problém se sekvencí EDID, která by nemohla být nastavena napevno, jelikož nemůže být zaručeno, jak velkým rozlišením bude připojený LCD displej disponovat. Na zařízení by bylo možné mít uloženou kolekci sekvencí EDID, která by pokrývala požadovaná rozlišení a mezi těmito sekvencemi by se na základně určitého vstupu vybrala správná sekvence. Příkladem vstupu by mohla být třeba kombinace propojení pinů na DPS. Pokud by daný hardware disponoval slotem pro SD kartu nebo USB portem, mohl by na tomto úložném zařízení očekávat konfigurační soubor, který by byl při inicializaci zdrojem konfiguračních dat pro daný hardware.

Bohatým zdrojem dat konfiguračních dat pro různé dostupné LCD displeje by mohla být knihovna FBTFT, která obsahuje rozsáhlou sadu ovladačů pro malé LCD displeje.

Výstupním zařízením však nemusí být pouze LCD displej. Pomocí SPI, či jiného rozhraní, by

se dal k takto vzniklému USB adaptéru připojit například LCD panel s patřičným hardwarovým vybavením a ovladačem a vytvořit tak zobrazovací jednotku rozsáhlých rozměrů.

Pro budoucí vývoj by jistě bylo vhodné veškerou funkcionalitu seskupit do jedné knihovny tak, aby byla universálně použitelná nejen na hardwaru od STMicroelectronics, ale prakticky na jakémkoli mikroprocesoru, jehož programová výbava může být vyvíjena v jazyce C. Tato knihovna by mohla být jednoduše importována a využita. Její obsah by tvořila funkcionalita zpracování dat, inicializační sekvence, sekvence EDID atp.

Jednoznačně je však zapotřebí zrychlit celý proces od přijímání dat po zobrazení. Ideální by bylo předělat komunikaci zařízení tak, ať pracuje s USB standardem High Speed. tento krok by výrazně zrychlil přenos dat a tím by se zvýšila i odezva LCD displeje. Dále by bylo vhodné provést refactoring a optimalizovat všechny algoritmy tak, aby jejich složitost byla co nejmenší a tudíž aby se prováděly co možná nejrychleji.

7 Využití v praxi

Výsledný produkt této diplomové práce otevírá značný počet směrů pro aplikaci, zdokonalení, rozšíření a využití v praxi.

Primárním cílem této diplomové práce bylo vytvořit LCD displej připojitelný přes rozhraní USB k počítači se systémem Linux. Velký potenciál využití tento LCD displej zajisté nalezne ve spojení s minipočítači, jako je například Raspberry Pi. Lze k němu sice připojit plnohodnotný monitor, televizi či projektor přes integrované a hojně rozšířené HDMI. Avšak tím, že se Raspberry Pi stalo velice populární nejen pro experimentování, ale také pro konkrétní řešení reálných problémů v praxi, nachází využití i v místech, kde může být obtížné připojovat velký monitor. Často uživatel ani nepotřebuje tak rozsáhlou zobrazovací plochu. Pro účely konfigurace nebo stručné indikace bohatě postačuje i malé zobrazovací zařízení, které je mobilní, lehce připojitelné, méně energeticky náročné ve srovnání s běžnými zobrazovacími jednotkami a navíc nezabere mnoho místa a hravě se vejde do kapsy, batohu či brašny.

Účel využití tohoto displeje může být i informativní a nemusí být využit jen pro konfiguraci, i když tento účel po něm může být považován jako vedlejší.

Takový modelový příklad může být třeba domácí meteo-stanice. Obecně by se dalo mluvit o jakémkoli automatizačním systému od řízení topení počínaje přes zabezpečovací zařízení až po automatizovanou obsluhu třeba akvária. Zaměřme se ale na zmiňovanou domácí meteo-stanici, která má v sobě implementovanou aplikaci, která obstarává směr dat jednotlivých senzorů nebo internetových zdrojů, pokud je k dispozici připojení k Internetu. S uživatelem pak tato aplikace vizuálně komunikuje právě pomocí takového malého displeje a poskytuje mu například přehledně informace nejen o teplotě, vlhkosti ale díky tomu že se jedná o grafický LCD displej, může zobrazovat třeba i radarové snímky srážek a oblačnosti či fotografie z on-line dostupných meteorologických serverů.

Výsledek této diplomové práce by se také mohl stát výchozím bodem nebo přinejmenším zdrojem informací pro uživatele, kteří chtějí experimentovat s vytvářením zobrazovacích jednotek, jelikož shrnuje poznatky o řešení a komunikaci technologie DisplayLink pro open-source operační systém Linux.

Výsledek práce taktéž může usnadnit vývoj a ulehčit analýzu zdrojových kódů ovladačů **udlfb** a **libdlo** vývojáři, který by projevil o tento směr zájem. Web autora ovladače, který je zmíněn jako druhý je již delší dobu neaktualizovaný a ovladač **udlfb** používá mírně modifikované algoritmy, než jsou ty které Florian Echtler, autor, zmiňuje.

Samozřejmě může tento projekt dále sloužit i jako ukázka propojení a spolupráce dostupných hardwarových periferií na vývojovém modulu od ST Microelectronics STM32F429I DISCOVERY, jelikož je zde využito jak komunikace přes USB rozhraní, tak integrovaného LCD displeje, dostupné SDRAM a dalších periferií. Tím dokazuje že tyto vývojové kity jsou velice vhodná jako hardwarový základ pro projekty podobného typu.

Záleží jen na fantazii a potřebách konkrétního uživatele či vývojáře, jak by vzniklé softwarové a technologické řešení využívající tento modul mohl využít pro řešení svého problému a nasadit jej do svého vlastního systému.

8 Závěr

Cílem diplomové práce bylo vytvořit malý LCD displej připojitelný přes USB rozhraní k počítači se systémem Linux. V rámci této práce byla před začátkem také provedena důkladná analýza, která měla za úkol vyhodnotit rizika, užitečnost a použití jednotlivých možných a také dostupných řešení. Bylo zvolena technologie DisplayLink, která se jevila jako nejvhodnější a nejperspektivnější. Analýzou se zabývá kapitola 2.

Na základě rozhodnutí z předcházející analýzy byla důkladně prozkoumána a nastudována technologie DisplayLink. V této fázi bylo využíváno všech dostupných materiálů o DisplayLink. Vzhledem k tomu že se nedjná o opensource řešení s primárním zaměřením na systémy Microsoft Windows, bylo vycházeno především z práce Floriana Echtlera, který se zabýval reengineeringem a vytvořil tak ovladač pro Linux nazvaný **libdlo**. Tento produkt a také ovladač **udlfb** (vydaný společností DisplayLink po zveřejnění libdlo) byly analyzovány a bylo z jejich částí vycházeno při implementaci.

Během analýzy také byl vybrán nejvhodnější hardware pro finální řešení. Tím je vývojový kit STM32F429 DISCOVERY. Podrobně jsou hardwarové součásti popsány v kapitole 3.1.

Po analýze následovala implementace, která se skládala z průběžného odposlouchávání datové komunikace po USB rozhraní mezi počítačem a originálním USB LCD displejem Lenovo LT1421. Během implementace byly průběžně studovány podrobnosti USB rozhraní, k vytvoření USB displeje na vývojovém kitu.

V průběhu implementace byla objevena úskalí, která značně ztěžují vytvoření takového LCD displeje pro více platforem než jen pro Linux. Jedním takovým úskalím je například šifrování dat při komunikaci 4.9, druhým je komprese **??**. Tyto dva procesy nejsou na Linuxu implementovány, tudíž jsou v této práci zmíněny teoreticky, nikoliv však jako součást implementace.

USB displej, který vznikal byl průběžně testován, respektive jeho části byly průběžně testovány, jak je popsáno v kapitole 5.

Pro reálně využití tohoto LCD displeje v praxi je však stále nutné provést optimalizaci všech algoritmů a poté řádně celé řešení otestovat v různých situacích.

Mezi hlavní přínosy této práce lze uvést především rozšíření znalostí v oboru vývoje USB zařízení a celkové funkcionality tohoto rozhraní. Dále rozšíření znalostí v oboru zobrazování, zpracování obrazových dat a vývoje embedded projektů.

Na zdokonalení výsledku této diplomové práce je zamýšleno pokračovat dále, případně zdrojové kódy zveřejnit na veřejném úložišti, aby umožnilo i případným dalším zájemcům snadněji pochopit a vyvinout vlastní zobrazovací zařízení přes rozhraní USB. Jednotlivé možnosti dalšího vylepšení a rozšíření celého řešení jsou rozebrány v kapitole 6.

Literatura

[1] RASPBERRY PI FOUNDATION. *Raspberry Pi* [online]. 2015 [cit. 2015–12–01] Dostupné z: http://www.raspberrypi.org.

[2] STMICROELECTRONICS. *32F429IDISCOVERY – Discovery kit with STM32F429ZI MCU* [online]. 2015 [cit. 2015–12–01] Dostupné z: http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF259090.

[3] RASPBERRY PI FOUNDATION. *Downloads* [online]. 2016 [cit. 2016–04–21] Dostupné z: https://www.raspberrypi.org/downloads/.

[4] LinkedIn Corporation - SlideShare. *Research in Internet of Things' Operating Systems (IoT OS's)* [online]. 24.01.2016 Salahuddin El-Kazak [cit. 2016–04–21] Dostupné z: http://www.slideshare.net/SalahuddinElKazak/research-in-internet-of-things-operating-systems-iot-oss.

[5] DISPLAYLINK. *Technology Overview* [online]. 2015 [cit. 2015–12–26] Dostupné z: http://www.displaylink.com/technology/technology_overview.php.

[6] DISPLAYLINK. *DisplayLink for Business* [online]. 2015 [cit. 2015–12–26] Dostupné z: http://www.displaylink.com/for-business/.

[7] HW server s.r.o. *USB 2.0 – Typy a formáty přenosů* [online]. Duben 2005 [cit. 2016–01–18] Dostupné z: http://vyvoj.hw.cz/navrh-obvodu/rozhrani/rs-485-rs-422/usb-20-typy-a-formaty-prenosu.html.

[8] STMICROELECTRONICS. *UM1734 User manual - STM32Cube USB device library* [online]. Květen 2015 [cit. 2016–01–19] Dostupné z: http://www.st.com/st-web-ui/static/active/jp/resource/technical/document/user_manual/DM00108129.pdf.

[9] LENOVO. *ThinkVision LT1421 14–inch Wide Flat Panel Monitor - Overview* [online]. Leden 2015 [cit. 2016–01–19] Dostupné z: https://support.lenovo.com/cz/cs/documents/pd015702.

[10] ECHTLER, F. – HODGES, C. *Reverse-Engineering DisplayLink devices* [online]. Listopad 2009 [cit. 2016–01–19] Dostupné z: https://events.ccc.de/congress/2009/Fahrplan/attachments/1431_paper.pdf.

[11] ECHTLER, Florian. *Startpage for Displaylink Stuff* [online]. Leden 2011 [cit. 2016–01–19] Dostupné z: http://floe.butterbrot.org/displaylink/doku.php.

[12] USBlyzer. *USBlyzer* [počítačový program]. Verze 2.1. Březen 2014 [cit. 2016–01–19] Dostupné z: http://http://www.usblyzer.com/. Software–based USB Analyzer and USB Data Traffic Sniffer for Widnows. 33–denní trial verze.

[13] THESYCON. *Thesycon USB Descriptor Dumper* [počítačový program]. Verze 1.84. Prosinec 2015 [cit. 2016–01–19] Dostupné z: `http://www.thesycon.de/eng/usb_descriptordumper.shtml`. Windows utility that displays the USB descriptors of any USB device.

[14] WIRESHARK FOUNDATION. *Wireshark* [počítačový program]. Verze 2.0.1. 29. prosince 2015 [cit. 2016–02–06] Dostupné z: `https://www.wireshark.org/#download`. The world's foremost network protocol analyzer.

[15] MOŃ, Tomasz. *USBPcap* [počítačový program]. Verze 1.0.0.7. [cit. 2016–02–06] Dostupné z: `http://desowin.org/usbpcap/`. Open-source USB sniffer for Windows. Wireshark plugin.

[16] AXELSON, Jan. *USB complete everything you need to develop custom USB peripherals* 3rd ed. Madison, WI: Lakeview Research, 2005. ISBN 1931448035..

[17] Extron Electronics. *Understanding EDID – Extended Display Identification Data* [online]. ExtroNews, 2009 [cit. 2016–02–08] Dostupné z: `http://www.extron.com/download/files/articles/understandingedid.pdf`.

[18] DELTACAST. *E-EDID Editor* [počítačový program]. Verze 1.31. 2015 [cit. 2016–02–08] Dostupné z: `http://www.deltacast.tv/products/free-software/e-edid-editor`. Editor datových sekvencí EDID .

[19] DISPLAYLINK. *How to install DisplayLink software on Ubuntu* [online]. 2015 [cit. 2016–03–01] Dostupné z: `http://support.displaylink.com/knowledgebase/articles/684649-how-to-install-displaylink-software-on-ubuntu`.

[20] GitHub. *notro/FBTFT Wiki* [online]. 2016 [cit. 2016–04–09] Dostupné z: `https://github.com/notro/fbtft/wiki`.

[21] USBTIPS. *What is the USB Enumeration Process?* [online]. Vincent Clarke, 2013 [cit. 2016–04–12] Dostupné z: `http://www.usbtips.com/what-is-the-usb-enumeration-process/`.

Struktura přiloženého CD

Součástí této diplomové práce je CD, které obsahuje elektronickou podobu tohoto textu ve formátu PDF a jeho zdrojový kód pro typografický systém LaTeX. CD dále obsahuje zdrojové kódy, fotografie a návody na konfiguraci. Adresářová struktura je následující:

Manuals návody na konfiguraci, informace, doplňkové texty

Photos fotografie

Print elektronická podoba diplomové práce ve formátu PDF

Source zdrojové kódy

ThesisSrc zdrojový kód diplomové práce pro LaTeX